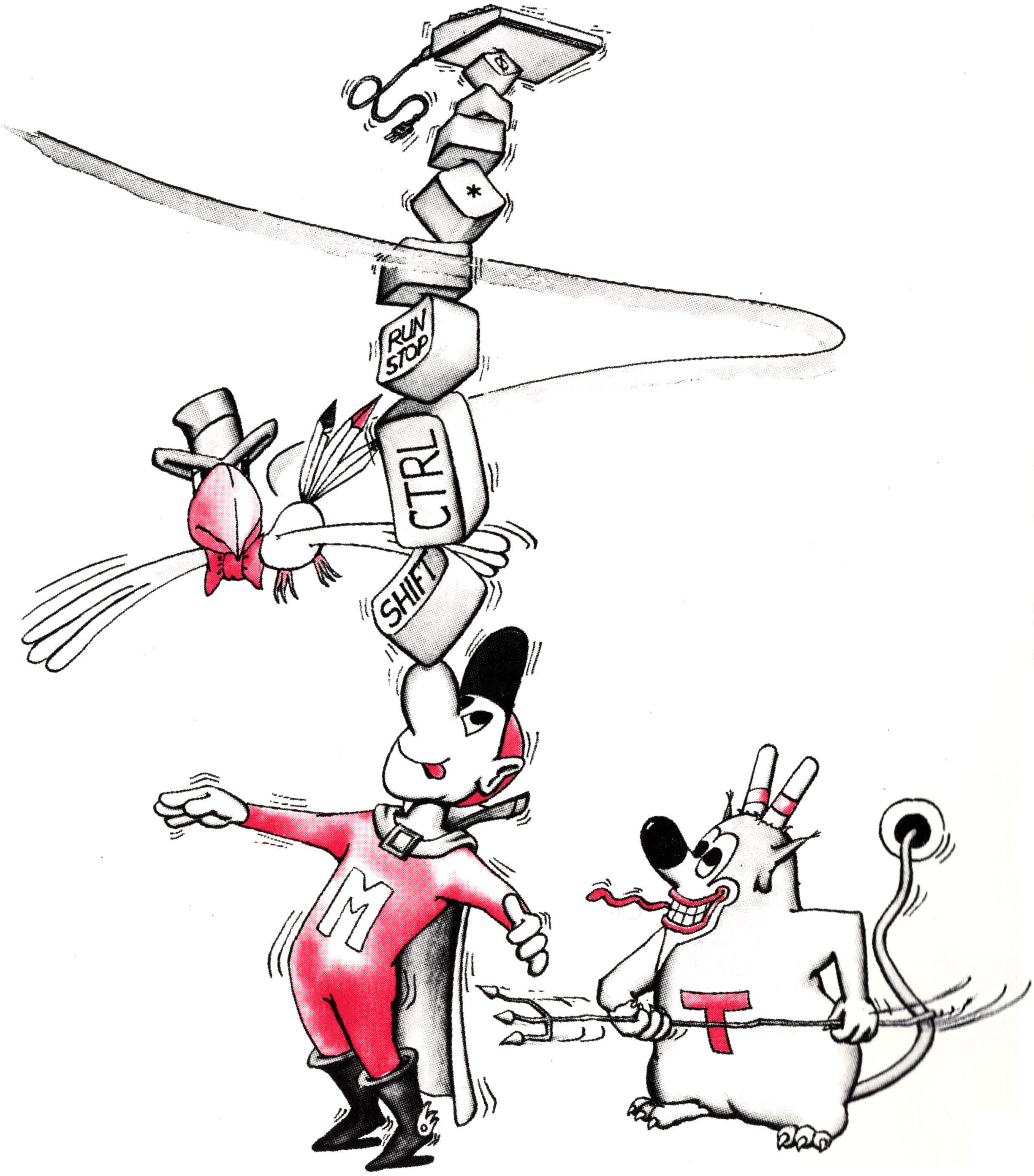
Computer kinderleicht

Lernen, spielen, programmieren

mit Lernspielen
für den C64

DM 16,80

Schneider-
Buch


RUN STOP
CTRL
SHIFT

Computer kinderleicht

Lernen, spielen, programmieren

Georg Seeßlen
Peter Fritzsche

Schneider-Buch

Inhalt

Vorwort 5

Welche Sprache spricht unser Computer? 8

Rechenmampfer – Spielanleitung 44

Der Super-Rechner
und das Geheimnis der Stellvertreter 46

Tips & Tricks für das Eingeben der Listings 86

Was heißt was in BASIC 89

Ballonjagd – Spielanleitung 94

Ballonjagd – Listing 97

Rechenmampfer – Listing 104

Textmon 64 – Anleitung 112

Textmon 64 – Listing 117

Liste der Steuerzeichen für den C 64 127

Listing-Seiten für
selbstgeschriebene Programme 128

Sinnbilder zum Entwerfen
eigener Programme 145

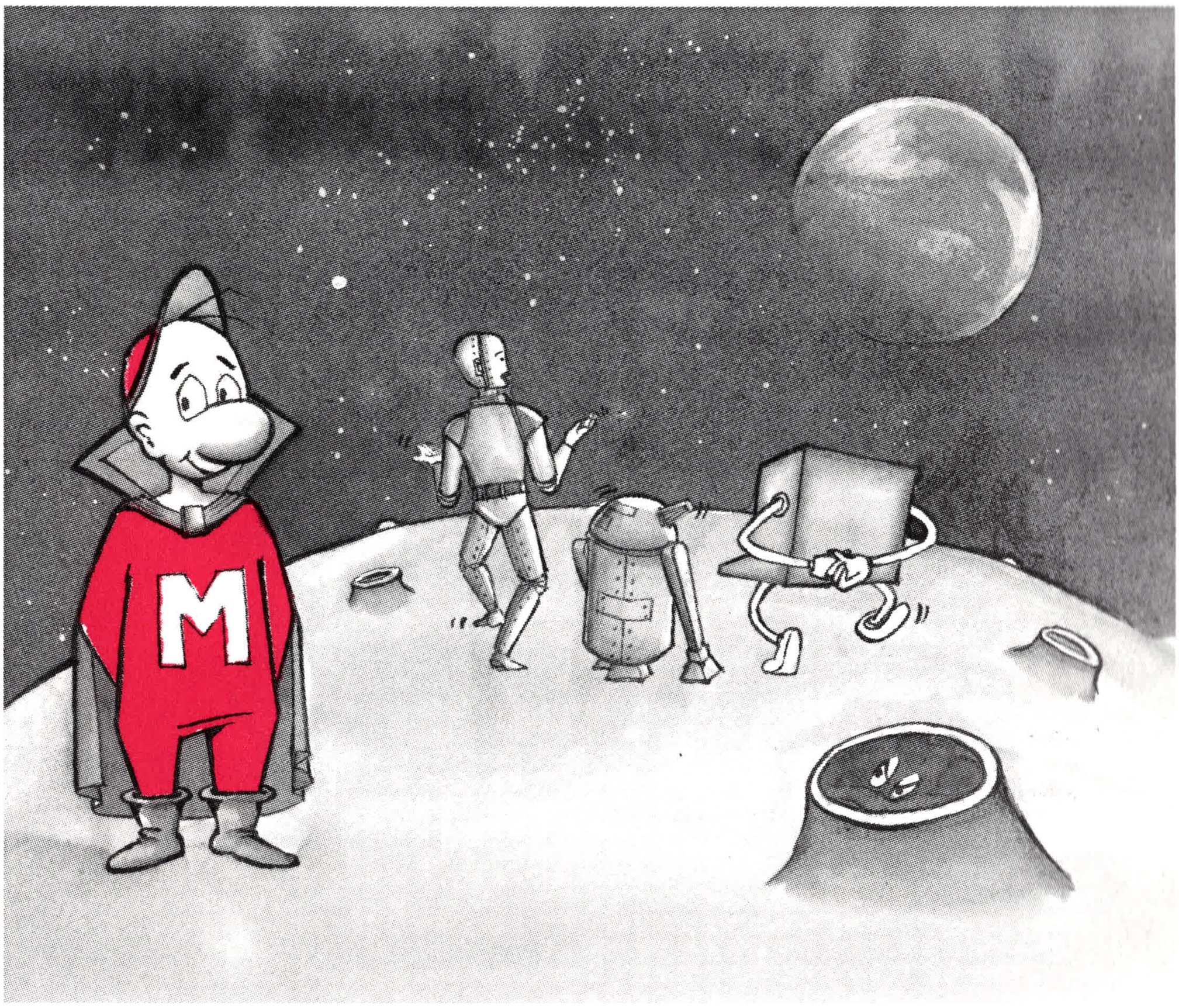
Vorwort

Wenn du dieses Buch in die Hand nimmst, hast du ja schon ein paar erste Computer-Erfahrungen gemacht. Entweder mit deinen Eltern zusammen oder mit Freunden oder einfach so für dich. Oder natürlich mit unserem ersten Buch.

Du weißt vielleicht schon genau, wie man die einzelnen Geräte miteinander verbindet. Im Handbuch findest du bestimmt die entsprechenden Anschlußhinweise.

Du kennst – wenigstens zu einem großen Teil – schon die Tasten auf der Tastatur deines Computers. Und du weißt, daß dieses Keyboard (gesprochen Kibohrd) der Teil des Computers ist, mit dem wir ihm etwas mitteilen können. Indem wir die richtigen Tasten nacheinander drücken, sagen wir ihm, was er tun soll.

PRINT „MAX“ heißt zum Beispiel, der Computer soll das Wort „Max“ auf den Bildschirm schreiben. Und PRINT 3–2 sagt dem Computer, er soll uns das Ergebnis von 3 minus 2 zeigen.



Max: Meine EDV-Anlage spielt am liebsten Weltraumspiele!

Wir geben unserem Computer Daten ein. Das sind Buchstaben, Zahlen und Zeichen, wie zum Beispiel Punkt oder Fragezeichen. Und wir sagen ihm, was er mit diesen Daten machen soll. Der Computer verarbeitet unsere Daten.

Was unser Computer macht, sobald wir mit ihm arbeiten, ist nichts anderes als *Datenverarbeitung.* Weil unser Freund ja alles elektronisch macht, nennt man das Elektronische Daten-Verarbeitung. Abgekürzt heißt das EDV.

Statt zu sagen: „Ich geh ein bißchen computern!", könntest du ebenso sagen: „Ich beschäftige mich etwas mit meiner EDV-Anlage."

Welche Sprache spricht unser Computer?

Die Sprache, in der wir mit unserem Computer sprechen, heißt BASIC. Das ist die gebräuchlichste Programmiersprache für Heimcomputer. Man kann ihnen auch andere Programmiersprachen beibringen, wie zum Beispiel LOGO oder FORTH.

BASIC ist die Abkürzung für einen ellenlangen englischen Ausdruck. Der heißt "**B**eginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode". Auf deutsch bedeutet das: Diese Programmiersprache kann auch von Anfängern gelernt werden, sie ist für fast alle Anwendungsmöglichkeiten der Computerei geeignet, und man verwendet symbolische Befehls-Wörter, um dem Computer zu sagen, was er tun soll.

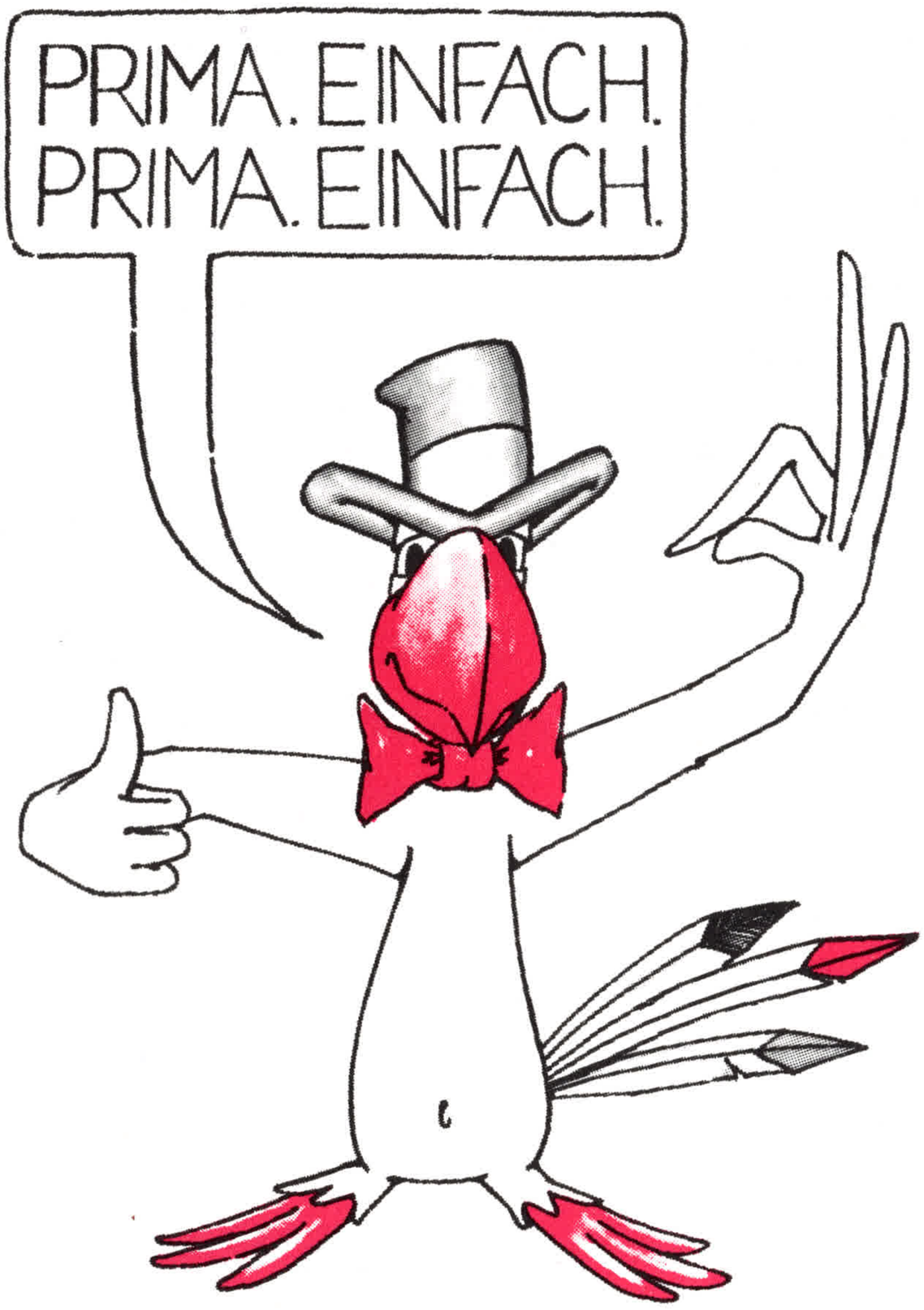
Die Anfangsbuchstaben dieses englischen Ausdrucks ergeben den Namen BASIC. Das englische Wort „basic" wiederum bedeutet soviel wie etwas Grundlegendes, das Wichtigste, aber auch etwas sehr Einfaches.

Man hat diese Programmiersprache an einer amerikanischen Universität entwickelt. Die Studenten sollten es leichter haben, den Umgang mit Computern zu lernen. Damals waren Computer noch gewaltig groß und sündhaft teuer. Niemand wäre auf die Idee gekommen, daß eines Tages kleine Homecomputer so viel leisten würden wie diese Rechenanlagen. Außerdem war der Umgang mit Computern

noch eine Sache für Spezialisten und gar nicht so selbstverständlich wie heute.

Am Anfang war BASIC (gesprochen „Bejßik“) nur etwas zum Lernen und Üben. Man kam mit sieben oder acht „Wörtern“ aus. Bald merkte man, daß man diese Computersprache ausbauen konnte. Immer noch ein Wort und noch ein Wort kamen hinzu. Es entwickelten sich auch verschiedene „Dialekte“ der BASIC-Sprache, die von den verschiedenen Computern verstanden wurden. In manchen von diesen Dialekten gibt es mittlerweile etwa hundert Worte.

Nun interessierten sich nicht mehr nur die Anfänger und Studenten für BASIC. Man merkte, daß sich in dieser Sprache die tollsten Programme schreiben



ließen. Sie waren zwar nicht so schnell und so raffiniert wie in anderen Sprachen, aber dafür mußte man kein zweiter Albert Einstein sein, um sich mit einem Computer zu verständigen. Und wenn man sich so ein BASIC-Programm ansieht, läßt sich mit einiger Übung bald erkennen, was es „leisten“ soll.

Das BASIC ist in erster Linie die Computersprache für die kleinen Computer zu Hause oder im Büro. Es gehört zu den sogenannten „höheren“ Computersprachen. Nicht etwa, weil BASIC schwierig zu lernen wäre. Es ist nämlich in Wirklichkeit schon etwas wie eine Übersetzung aus der Sprache, die der Computer am einfachsten und am schnellsten versteht, der Maschinensprache.

Man kann sich gut vorstellen, daß BASIC eine Sprache ist, die einerseits die Maschine, nämlich dein Computer, versteht, und andrerseits auch der Mensch, nämlich du.

Unser Vorteil ist, daß der Computer, mit dem wir arbeiten, sein BASIC schon gelernt hat. Er hat sozusagen einen kleinen Übersetzer fest eingebaut. Falls wir eine andere Programmiersprache verwenden wollen, müssen wir unseren Computer erst mit einem entsprechenden Programm „füttern“.

Der eingebaute Übersetzer braucht eine gewisse Zeit zum Übersetzen. Das ist zwar tausendmal weniger, als ein menschlicher Übersetzer brauchte, aber immerhin doch ein bißchen. Viele von den Video- und Computer-Spielen, die du kennst, sind deswegen direkt in Maschinensprache geschrieben.

In der BASIC-Sprache gibt es nicht nur verschiedene Dialekte (1). Für die Fortgeschrittenen hat man auch noch die eine oder andere Erweiterung geschaffen. Das „Extended BASIC“ zum Beispiel, was nichts anderes als „erweitertes BASIC“ heißt – „Simon's BASIC“, das sich besonders gut für Grafik und Musik eignet – oder COMAL, wo man versucht hat, die Vorteile von BASIC und LOGO zusammenzubringen. Dabei kommen meistens nur ein paar Befehlswörter dazu, für die man im „normalen“ BASIC die eine oder andere Zeile vollschreiben müßte.

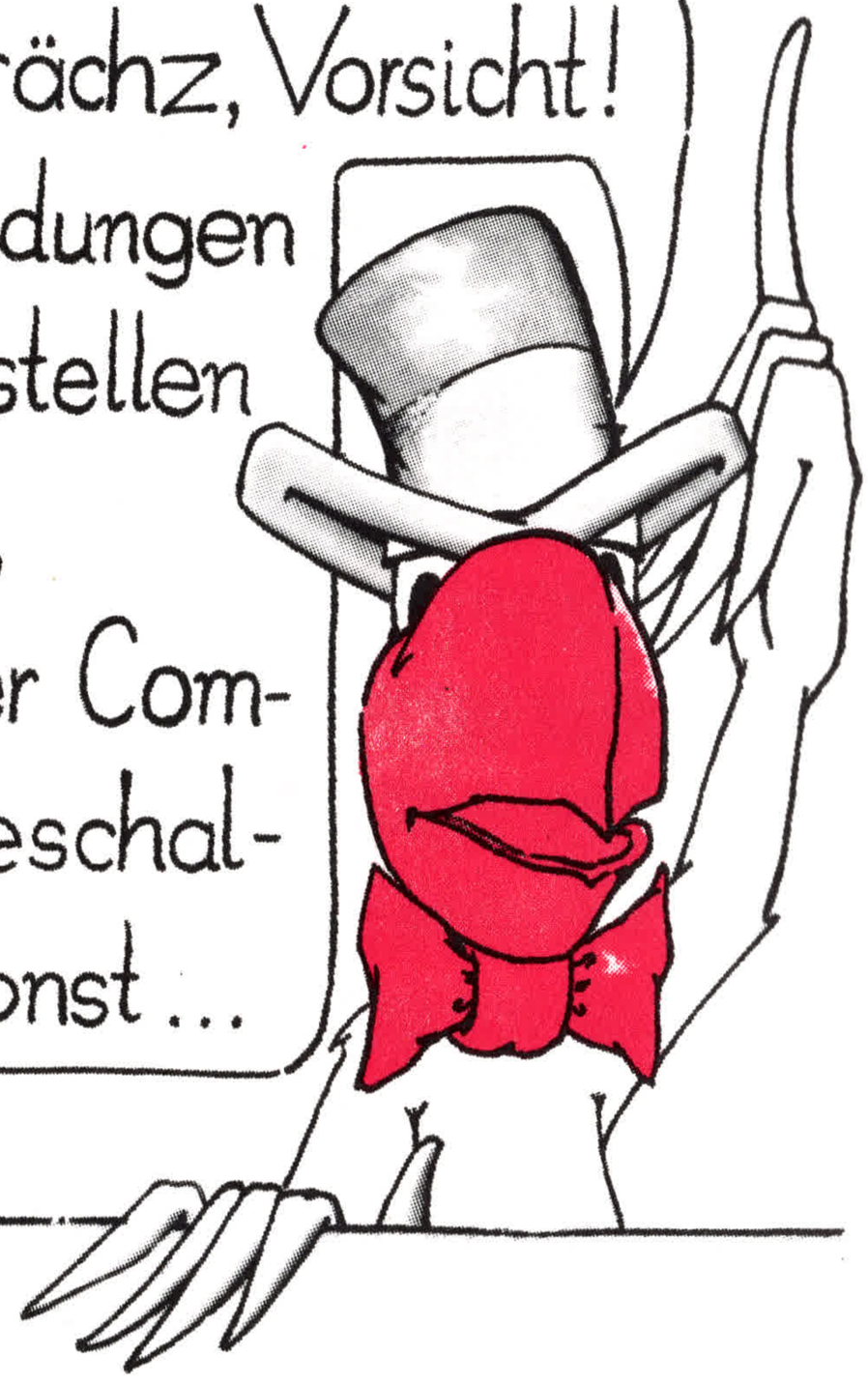
Allerdings kommt man anfangs ganz gut ohne ein erweitertes BASIC aus. All diese Möglichkeiten erwähne ich hier lediglich, damit du weißt, was gemeint ist, wenn da und dort die Rede davon ist.

So, und nun reicht es vorerst mit den Erklärungen. Jetzt wollen wir (endlich!) wieder am Computer selber üben. Genauer gesagt: Wir geben unserer EDV-Anlage in der Programmiersprache BASIC Anweisungen und Befehle, damit sie unsere Daten verarbeitet.

Sind Computer und Bildschirm richtig eingestellt? Dann kann es ja losgehen.

Das wichtigste beim Programmieren kennst du ja bereits. In einem BASIC-Programm muß dem Computer eine „Botschaft“ nach der anderen übermittelt werden. Dazu schreiben wir das Programm in Zeilen.

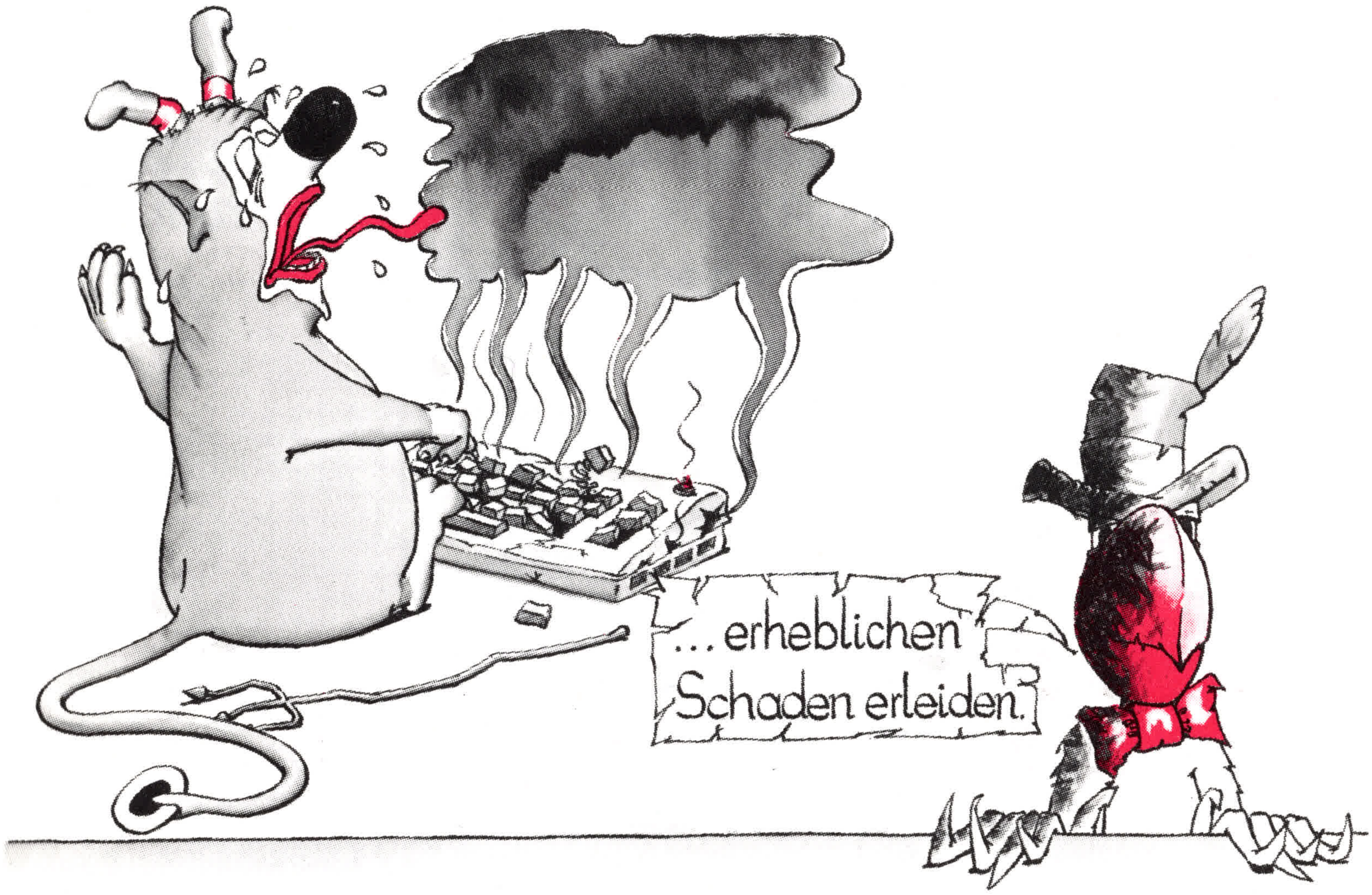
Achtung, Krächz, Vorsicht! Kabelverbindungen niemals herstellen oder lösen, solange der Computer angeschaltet ist! Sonst...



... sonst Könnte der Computer erheblichen Schaden....

Ratsch!

PENG!!!



Vor jede Zeile setzen wir eine Nummer. Vorläufig schreiben wir in jede Zeile nur einen Befehl.

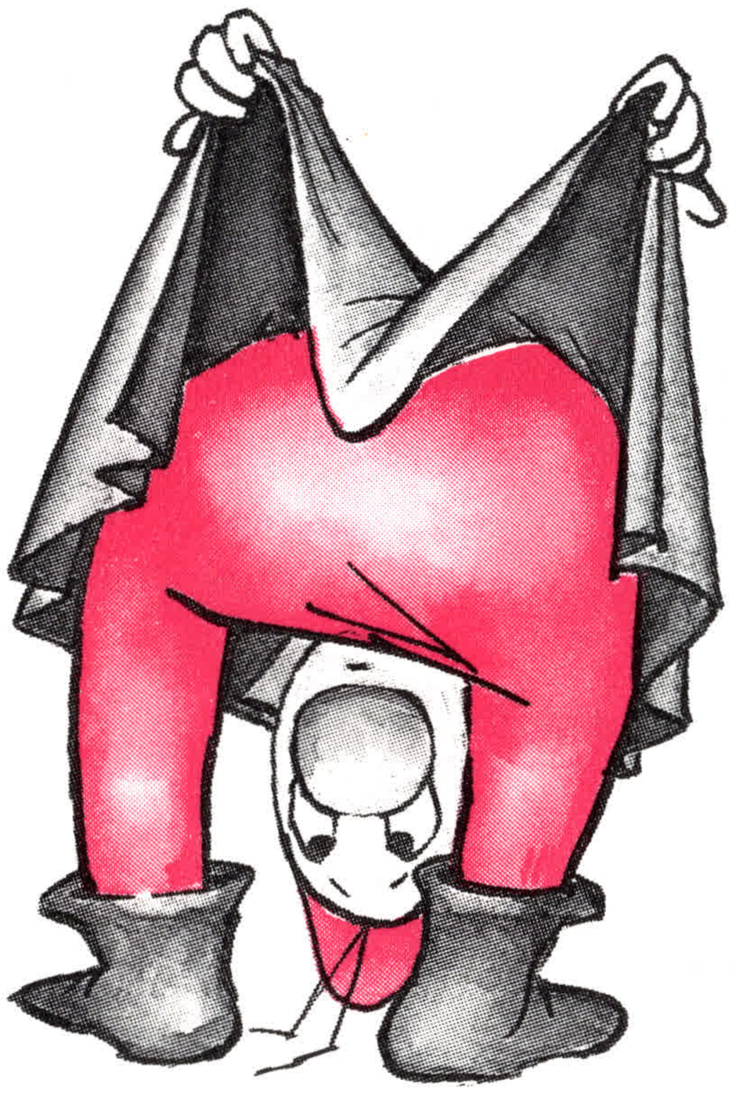
Weißt du noch, in welchen Schritten wir die Zeilen numerieren? Einerschritte? Zehnerschritte? Hunderterschritte?

Schau dir das kleine Programm auf der nächsten Seite an, wenn du nicht ganz sicher bist.

Vielleicht hast du Lust, dieses Programm gleich nachzutippen. Es ist nicht schwer. Achte nur darauf, daß hinter dem ersten Anführungszeichen fünf Leerzeichen (die Balkentaste!) und zwischen den beiden „Kästen“ drei Leerzeichen sind.

Nachdem du die Zeilen mit den PRINT-Befehlen geschrieben hast, kommt Zeile 60. Sie sagt dem Computer, er soll wieder zur Zeile 10 gehen und dort

mit der Ausführung des Programms weitermachen. Vielleicht erinnerst du dich daran, daß wir schon im ersten Band mit diesem GOTO-Befehl Bewegung in unser Computer-Bild gebracht haben – zum Beispiel bei unserem fliegenden Pfeil.)



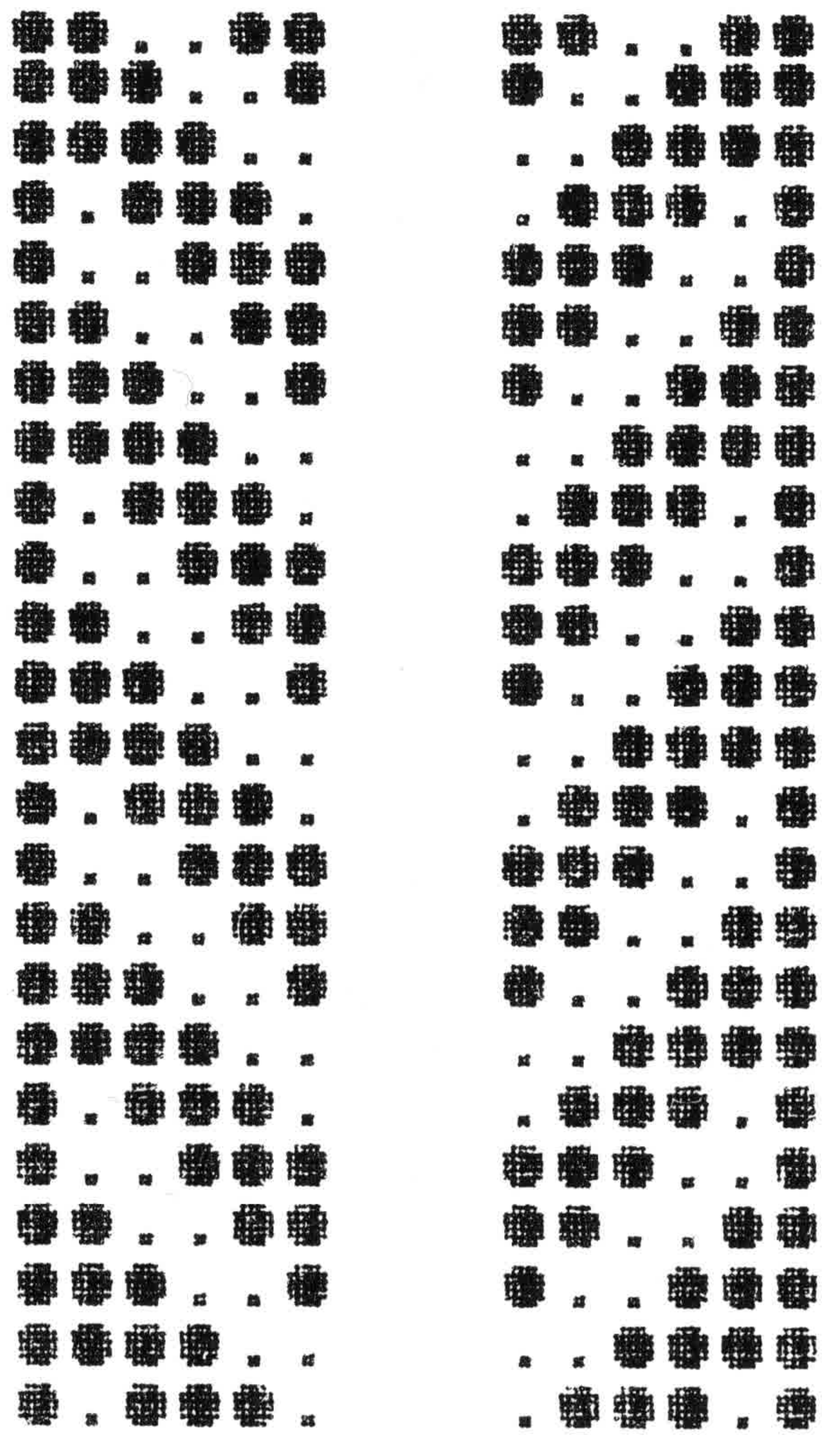
```
READY.

60 GOTO10
50 PRINT"
40 PRINT"
30 PRINT"
20 PRINT"
10 PRINT"
```

Nachdem du das Programm mit

RUN

gestartet hast, erscheint auf dem Bildschirm eine

bewegte Grafik. Das sieht ein wenig wie ein Tannenbaum aus, der dauernd nach oben wächst. Oder was siehst du?



Darin steckt aber noch ein kleiner optischer Trick. Beobachte mal unseren Computerfilm eine Zeitlang ganz konzentriert! Vielleicht bemerkst du schon einige Veränderungen. Zuerst könnte man denken, unser „Baum“ (oder was man sich sonst darunter vorstellt) wird nach oben hin schmaler. Daß das nicht stimmen kann, weißt du. Du hast die Säulen ja schließlich selbst programmiert. Noch eindrucksvoller ist es, daß die Bewegung nach einer Weile konzentrierten Schauens plötzlich nicht mehr nach oben zu gehen scheint, sondern waagrecht nach innen.

Wollen wir uns jetzt an ein paar Beispielen noch einmal vergegenwärtigen, wie gut wir schon BASIC beherrschen?

Unser erstes Befehlswort hieß: PRINT. Damit läßt sich schon eine ganze Menge anstellen, wenn man es zum Beispiel *mit* und einmal *ohne* Anführungszeichen schreibt.

Machen wir eine kleine Übung anhand der Rechnung 3 mal 7. Was passiert, wenn wir folgendes schreiben?

```
10 PRINT 3*7
```

Natürlich gar nichts. Denn zuerst müssen wir ja mit

RETURN

dem Computer bestätigen, daß er diese Programmzeile tatsächlich in sein Gedächtnis aufnehmen soll. Außerdem müssen wir das Programm starten. Denn ein Programm ist unser

```
10 PRINT 3*7
```

auch wenn es nur aus einer einzigen Programmzeile besteht. Die Anweisung für den Start des Programms ist

```
RUN
```

und zur Bestätigung natürlich wieder

RETURN

Sobald du das Programm geschrieben und gestartet hast, müßte auf dem Bildschirm zu sehen sein:

```
10 PRINT 3*7
RUN
 21
READY.
```

Mit dem

READY

(weißt du ja noch) meldet der Computer, daß er bereit ist, neue Aufgaben zu übernehmen.

Jetzt wollen wir eine andere Zeile 10 schreiben. Dazu könnten wir das Programm erst einmal aus dem Gedächtnis des Computers löschen. Weißt du noch, wie das geht? Ein Tip: Wir wollen, daß der Computer etwas neu anfängt, und „neu“ heißt auf englisch NEW. Und dann könnten wir den Bildschirm „saubermachen“. Dafür haben wir eine eigene Taste, die man mit oder ohne

SHIFT

verwenden kann.

Versuche zuerst selbst, ob du das Gedächtnis deines Computers zuerst löschen und dann den Bildschirm saubermachen kannst.

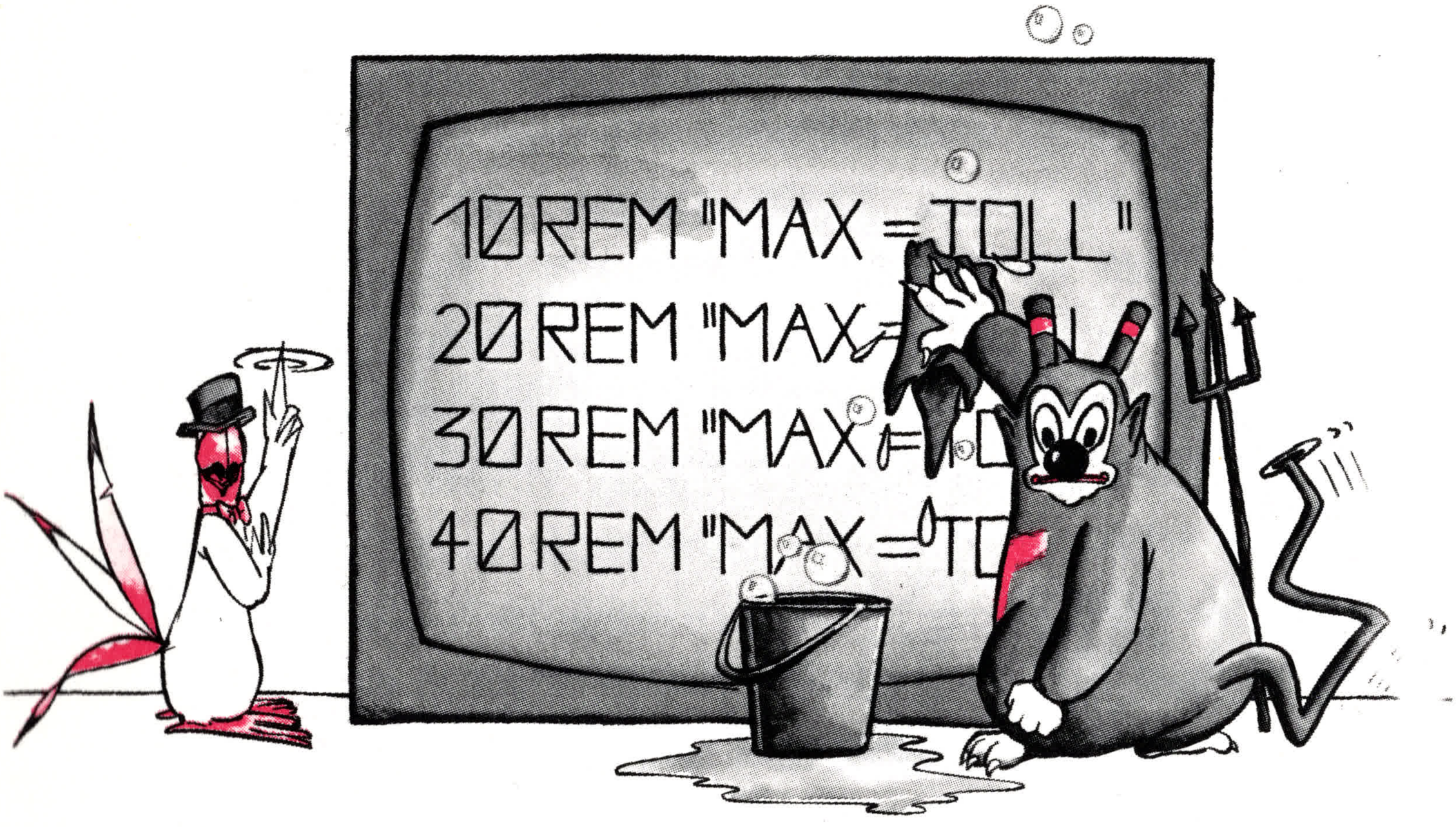
Eigentlich wäre das gar nicht nötig. Denn der Computer schreibt eine neue Zeile immer über die frühere drüber, falls sie dieselbe Nummer hat. Er löscht dabei die vorher geschriebene aus seinem Gedächtnis.

Das kann man ja überprüfen. Schreib zum Beispiel

```
10 PRINT 3+4
```
RETURN

und dann

```
10 PRINT 3*4
```
RETURN



Nun startest du das Programm mit

RUN

Was gibt der Computer als Ergebnis auf dem Bildschirm an? Klar, so sieht die Sache aus:

```
10  PRINT  3+7
10  PRINT  3*7
RUN
  21

READY.
```

Tatsächlich – von zwei Zeilen mit derselben Nummer gilt für den Computer immer die, die zuletzt eingegeben worden ist. Die erste Zeile mit der Nummer 10 ist also für ihn gestrichen. Das kannst du dir noch

mal verdeutlichen, indem du die beiden Zeilen mit der Nummer 10 noch einmal schreibst. Dazu gibst du dem Computer die Anweisung

LIST

Das heißt, er soll das ganze Programm noch einmal „auflisten“. Und zwar so, wie es im Augenblick gilt. Auf dem Bildschirm sieht das so aus:

```
10 PRINT 3+4
10 PRINT 3*4
LIST

10 PRINT 3*7
READY.
```

Auf dieselbe Weise können wir eine Zeile im Programm auch völlig auslöschen. Dazu schreiben wir sozusagen eine leere Zeile über eine andere, die überflüssig ist oder in der etwas Falsches steht. Dazu müssen wir jedoch nicht mit dem Cursor an die entsprechende Zeile zurück, sondern wir tippen am linken Bildschirmrand (weiter unten, wo der Bildschirm noch frei ist), die Zeilennummer ein und schreiben dahinter – nichts.
Also geben wir ein

1 Ø RETURN

Zunächst geschieht gar nichts. Jetzt versuche, das Programm aufzulisten.

Aha, auch nichts. Und starten kann man das Programm offensichtlich nicht mehr.

Der Computer hat die Programmzeile mit der Nummer 10 aus seinem Gedächtnis gelöscht. Mit einer Zeile, in der nichts steht, kann er nämlich nichts anfangen. Das ist wie ein Brief mit einem leeren Blatt Papier drin.

Die alte Zeile mit der Nummer 10 hat er trotzdem gelöscht. Wie wenn man in einem Brief einen Satz mit leerem Papier überklebt. Das kann sehr praktisch sein, falls man einmal Fehler gemacht hat und nicht auf das umständliche Radieren mit unserem schnellen Löscher, den Cursor-Tasten und der Leertaste zurückgreifen will.

Wir können es gleich ausprobieren. In dem folgenden kleinen Programm ist eine Zeile fehl am Platz. In ihr steht ein Begriff, der überhaupt nicht zu den anderen paßt.

```
10 PRINT "DREIECK"
20 PRINT "KREIS"
30 PRINT "PFANNKUCHEN"
40 PRINT "VIERECK"
```

Die unpassende Zeile läßt sich mühelos herausholen, und zwar ohne zu radieren. Und genauso leicht ist es, eine neue Zeile mit einem richtigen Begriff – zum Beispiel „Halbkreis“ – einzufügen. Wie das aussieht, kannst du dir noch einmal hier auf

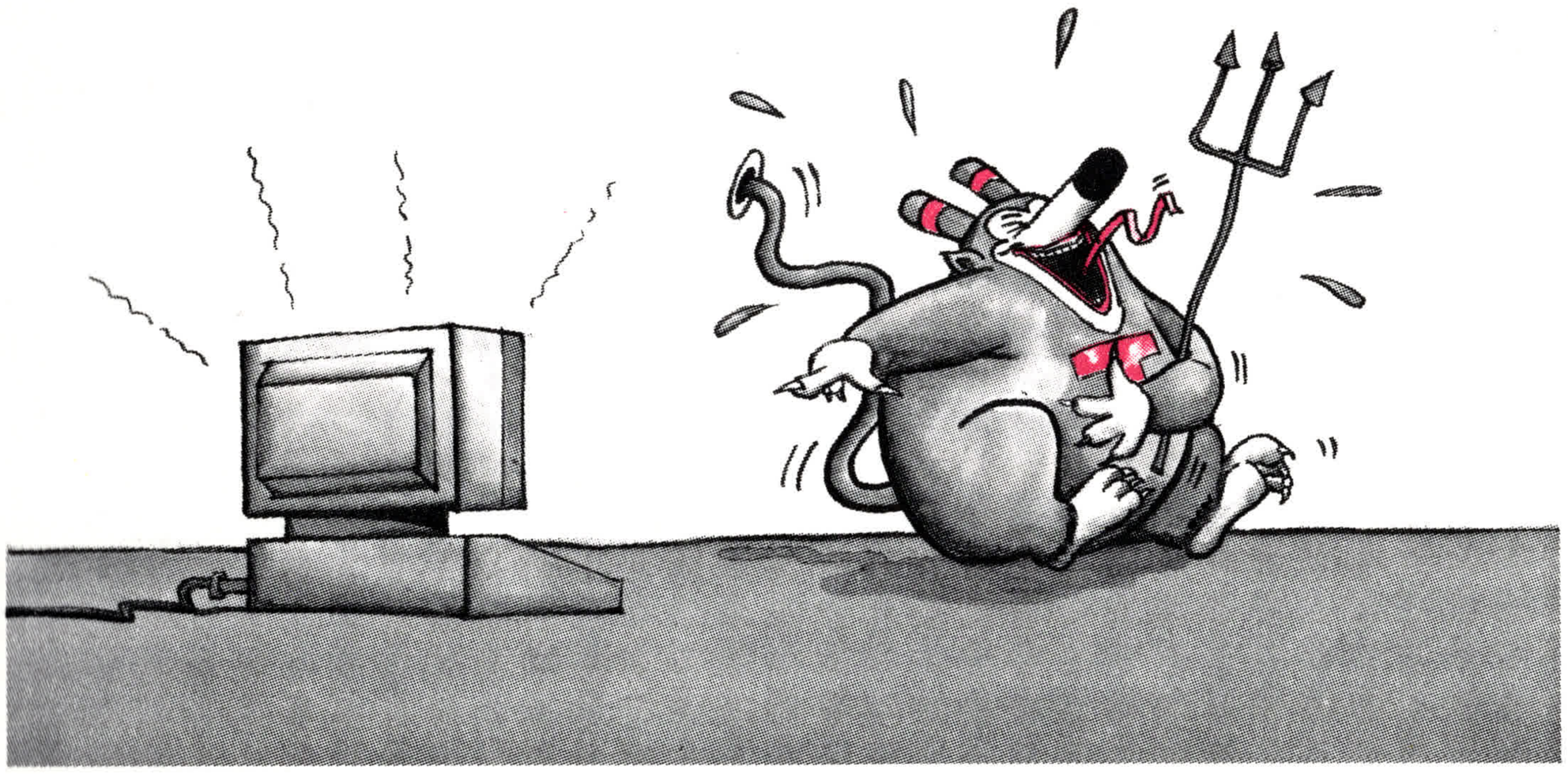
unserem Bildschirm anschauen. Aber sicher bist du mittlerweile selbst drauf gekommen.

```
10 PRINT "DREIECK"
20 PRINT "KREIS"
30 PRINT "PFANNKUCHEN"
40 PRINT "VIERECK"
RUN
DREIECK
KREIS
PFANNKUCHEN
VIERECK

READY.
30 PRINT "HALBKREIS"
LIST

10 PRINT "DREIECK"
20 PRINT "KREIS"
30 PRINT "HALBKREIS"
40 PRINT "VIERECK"
READY.
RUN
DREIECK
KREIS
HALBKREIS
VIERECK

READY.
```

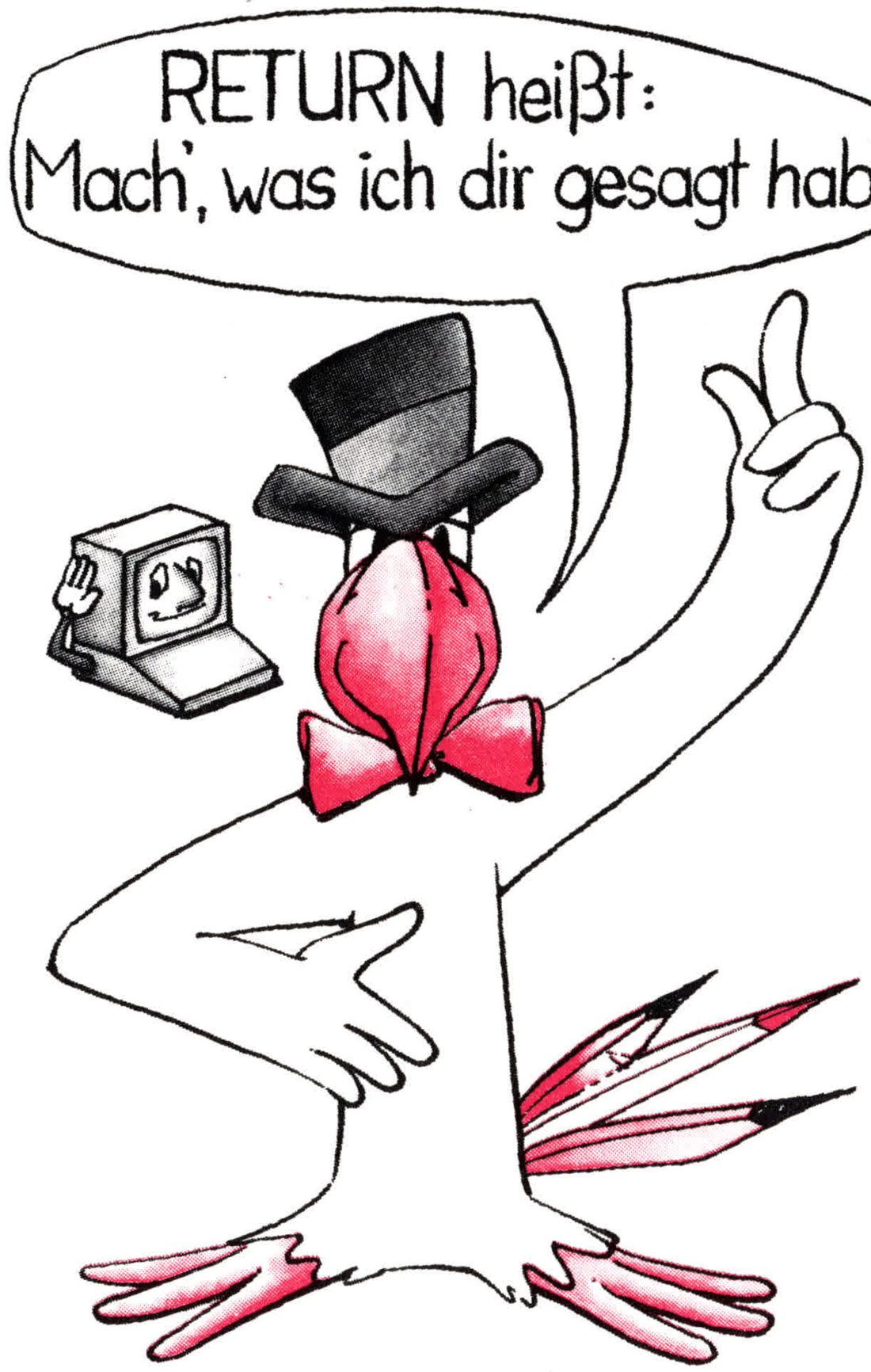
Das Verbessern durch „Darüberschreiben“ funktioniert selbstverständlich auch in einer einzelnen Zeile. Schauen wir uns dazu das nächste Programm an:

```
10 PRINT "1*15=15"
20 PRINT "2*15=30"
30 PRINT "3*15=46"
40 PRINT "4*15=60"
```

Da ist nun wieder eine Zeile drin, in der etwas Falsches steht. Lasse bitte beim Abtippen den Fehler drin, auch wenn du ihn gleich erkannt hast.



Diesmal ist nicht alles falsch, was nach dem PRINT-Befehl kommt, sondern nur eine einzige Zahl. Um so etwas zu berichtigen, schreiben wir nicht die ganze Zeile neu. Wir steuern nur unseren Cursor auf die entsprechende Stelle und tippen die richtige



Zahl über die falsche. (RETURN nicht vergessen, denn damit erst wird die Verbesserung ins Gedächtnis des Computers gebracht!)

Nachdem wir es noch einmal aufgelistet haben, sieht unser neues Programm folgendermaßen aus: (Vor Eingabe des LIST-Befehls gehst du mit dem Cursor am besten nach unten in eine freie Zeile, sonst gibt es leicht Fehler.)

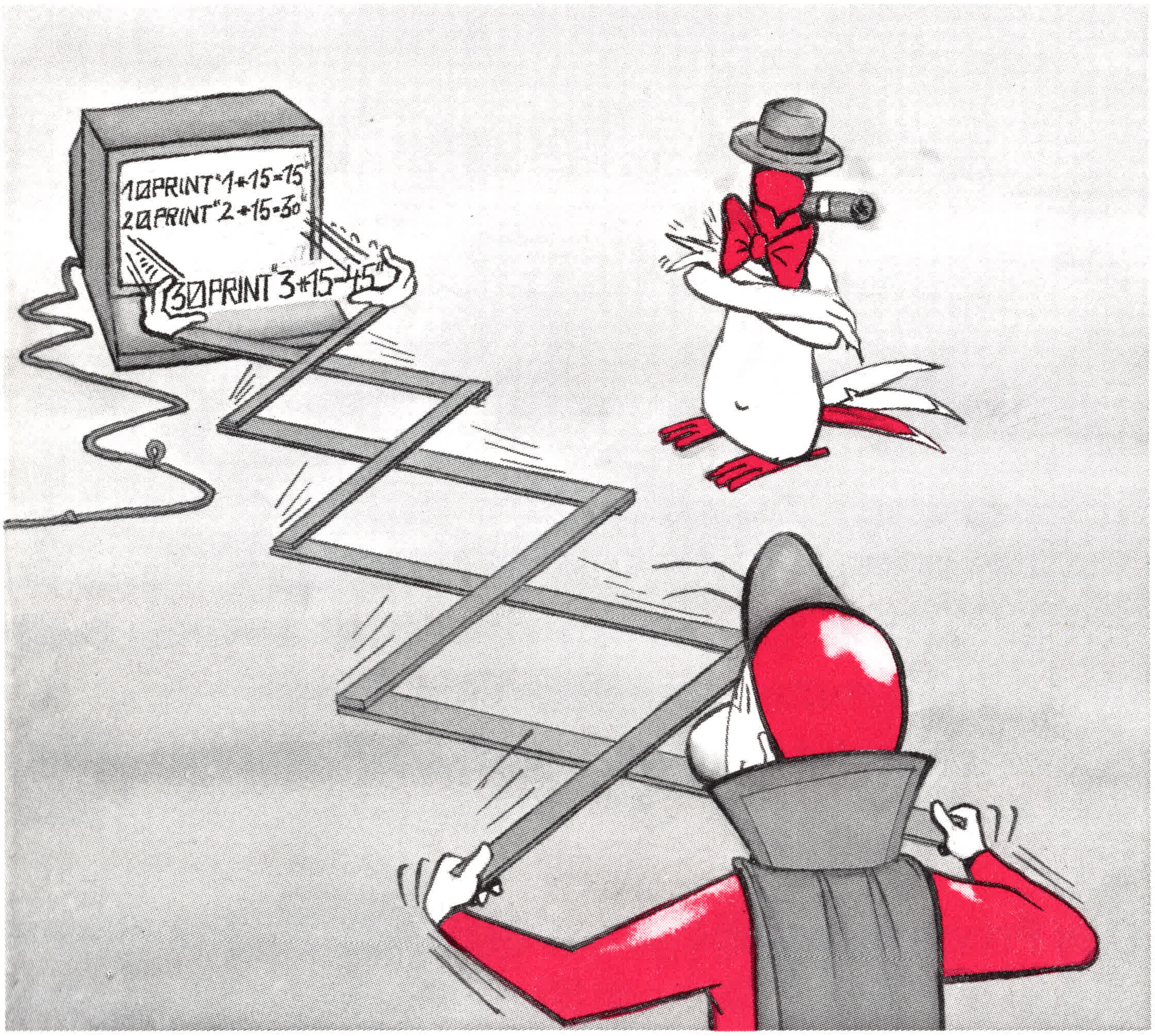
```
10 PRINT "1*15=15"
20 PRINT "2*15=30"
30 PRINT "3*15=45"
40 PRINT "4*15=60"
```

Falls die Programme umfangreicher und komplizierter werden, dauert es mit dieser Methode allerdings ein bißchen zu lange. Da ist es besser, man holt sich immer nur die eine Zeile aus dem Listing heraus, die man verändern möchte.

Wie das zu schaffen ist? Ganz einfach. Man verwendet die normale LIST-Anweisung und tippt dahinter die Nummer der Zeile ein, die man sich näher anschauen und vielleicht verbessern möchte. Mit

```
LIST 30
```

bekommen wir somit die Zeile 30 aus dem Listing. Hole dir mal einzelne Zeilen aus dem letzten Programm heraus!



Und auf eine ähnliche Weise kann man sich auch ganze Programm-Teile herauslisten lassen. Wir verwenden dazu den Strich (der ja auch unser Minus-Zeichen ist). So erhalten wir zum Beispiel mit der Anweisung

LIST 20–30

als Listing auf dem Bildschirm folgendes:

```
LIST20-30

20 PRINT"2*15=30"
30 PRINT"3*15=45"
```

Und was passiert bei der folgenden Anweisung?

LIST –20

Und bei dieser?

LIST 20–

Mach's wie der Max: Ausprobieren!

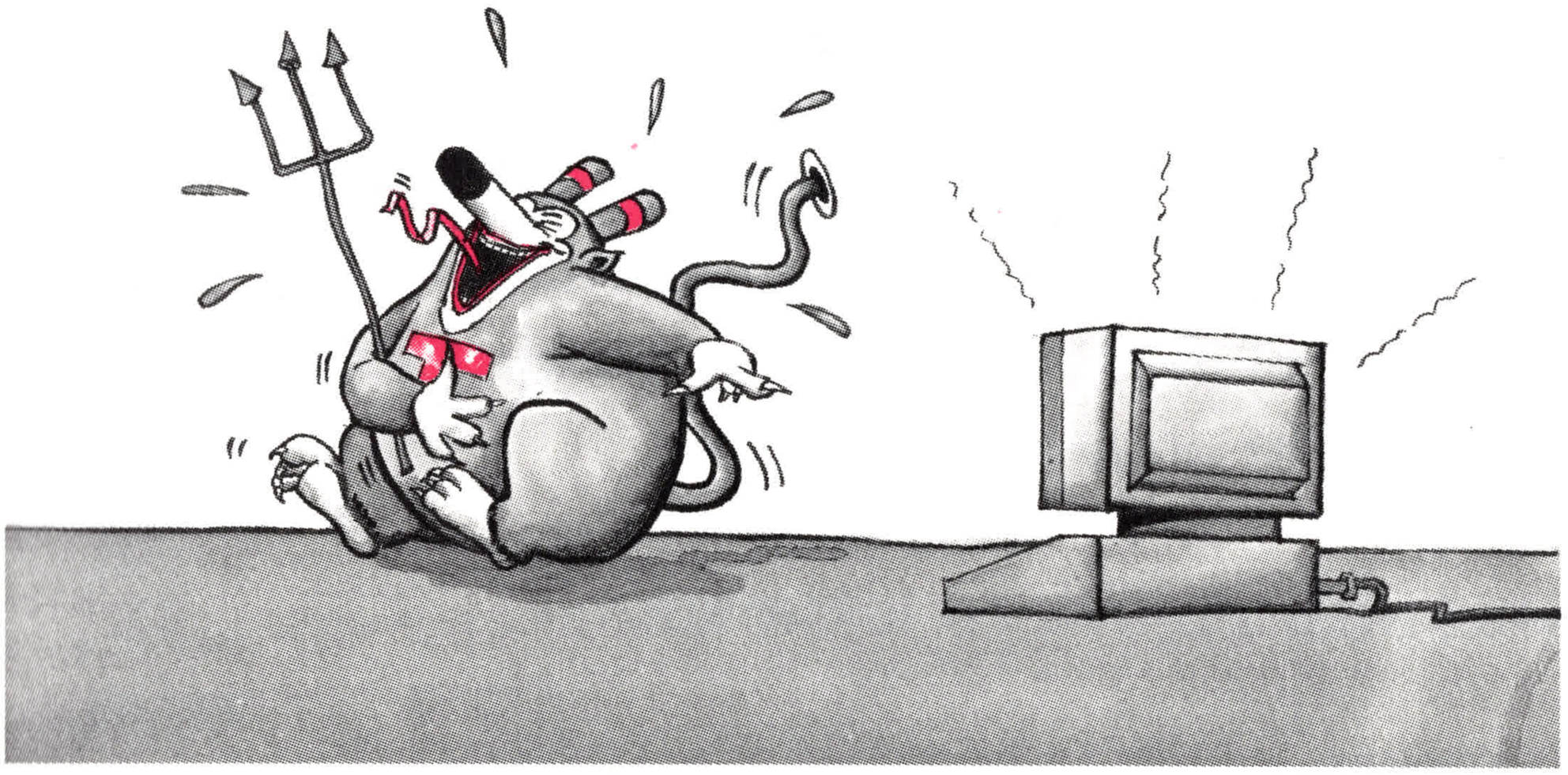
Auch starten kann man ein Programm nicht nur von der ersten Zeile ab, sondern auch ab einer beliebigen Zeile mitten im Programm. Tippe

RUN 20

Das Programm beginnt mit der Zeile 20 und arbeitet alle folgenden Zeilen ab. Das [–]-Zeichen kann man dabei nicht verwenden.

All dies können wir nun an einem Beispiel gründlich durchgehen. Hier ist ein kleines Programm, in dem ein paar Fehler stecken. Tippe es mitsamt den Fehlern bitte nach, oder lade es von der Diskette, falls du dieses zusammen mit einer Programmdiskette gekauft hast.

```
10 PRINT "OHNE RUHEN, OHNE RASTEN"
20 PRINT "HAUT DER MAX IN SEINE TASTEN"
30 PRINT "AUF DEM BILDSCHIRM, WIE IHR WI
SST"
40 PRINT "UNSINNSZEILE"
50 PRINT "STEHT MANCHMAL TOLLES"
60 PRINT "                MANCHMAL MIETZ"
```



Wenn du das Programm einmal startest, kommt dabei folgendes heraus:

```
RUN
OHNE RUHEN, OHNE RASTEN
HAUT DER MAX IN SEINE TASTEN
AUF DEM BILDSCHIRM, WIE IHR WISST
UNSINNSZEILE
STEHT MANCHMAL TOLLES
                MANCHMAL MIETZ

READY.
```

In diesem Programm ist eine Zeile gänzlich überflüssig. In einer anderen Zeile hat sich ein kleiner Fehler eingeschlichen. In einer weiteren Zeile gibt es einen falschen Zwischenraum, und ein Wort ist völlig danebengeraten.

Um dieses Programm zu verbessern, gibt es mehrere Möglichkeiten. Probiere ein bißchen. Selbst wenn du es schon weißt, kann ein wenig „Training“ nicht schaden. (Wir haben auf der Diskette ein paar solcher Fehlerprogramme untergebracht, die du

verbessern kannst.) Du kannst auch noch mit jemandem zusammen spielen und üben. Zum Beispiel schreibt der eine von euch ein solches Gedicht mit einer Unsinnszeile, und der andere holt sie heraus. Sie muß ja nicht immer so leicht zu erkennen sein wie in unserem Beispiel.

Jetzt aber zurück zu unserem

PRINT

Lösche bitte Bildschirm und Gedächtnis deines Computers. Nein, Max, nicht den Computer aus- und wieder einschalten! Das wollten wir uns doch als „Notbremse“ aufbewahren – für den Fall, daß sonst gar nichts mehr geht. Jetzt verwenden wir eine viel elegantere Methode! Wie du weißt, bedeuten die folgenden Befehle etwas ganz Verschiedenes:

PRINT „3*7“
PRINT 3*7

Man kann daraus ein Programm zusammensetzen. Es sieht so aus:

```
10 PRINT "3*7"
20 PRINT 3*7
```

In der ersten Zeile wird dem Computer gesagt, er soll die Rechenaufgabe ausdrucken. In der zweiten Zeile

errechnet er die *Lösung* und bringt sie auf den Bildschirm. Nach dem Start des Programms sieht es also auf dem Bildschirm so aus:

```
10 PRINT "3*7"
20 PRINT 3*7
RUN
3*7
 21

READY.
```

Noch schöner wird es, wenn wir hinter die Aufgabe ein Gleichheitszeichen setzen – es sieht so aus: [=] Auf dem Bildschirm siehst du jetzt:

```
10 PRINT "3*7="
20 PRINT 3*7
RUN
3*7=
 21

READY.
```

Schon ganz nett, oder? Nun gibt es aber noch einen Trick, diese Rechnung genau so zu schreiben, wie du sie normalerweise auch in dein Rechenheft schreibst, nämlich nacheinander in einer Zeile.

Dazu verwenden wir dieses Zeichen: [;]. Man nennt es auch in der Computerei „Semikolon".

In unserer BASIC-Sprache hat es mehrere Bedeutungen. Einmal ist das Semikolon ein ganz normaler Strichpunkt. Nämlich dann, wenn wir es in einem

Text verwenden. Doch das gilt nur, solange dieser Text in Anführungszeichen steht, zum Beispiel hinter einem PRINT-Befehl. Andererseits kann das Semikolon dem Computer auch etwas über den Abstand sagen, mit dem er mehrere verschiedene Sachen auf dem Bildschirm ausgeben soll.

Sobald nach einem PRINT-Befehl mehr als eine „Sache“ steht, die ausgegeben oder gelöst werden soll, dann müssen diese „Sachen“ durch ein Zeichen getrennt werden. Dazu verwenden wir das Semikolon.

Das Semikolon bewirkt, daß zwischen zwei „Sachen“, die „gePRINTet“ werden, *keine* Leerstelle steht. Probieren wir es aus: Das Programm

```
10 PRINT "3*7=";"4*7=";"5*7=";"6*7="
```

ergibt – mit RUN und RETURN gestartet –:

```
10 PRINT "3*7=";"4*7=";"5*7=";"6*7="
RUN
3*7=4*7=5*7=6*7=

READY.
```

Das ergibt natürlich keinen Sinn. Anders sieht es aus, wenn man gleich nach den Lösungen fragt. Das Programm

```
10 PRINT 3*7;4*7;5*7;6*7;7*7
```

läuft so ab:

```
10 PRINT 3*7;4*7;5*7;6*7;7*7
RUN
 21   28   35   42   49

READY.
```

Die beiden Leerzeichen zwischen den jeweiligen Lösungen kommen daher, daß der Computer bei Lösungszahlen automatisch ein Leerzeichen davor und eins dahinter läßt.

Diese beiden Dinge, die Aufgabe und die Lösung, können wir nun auch in *einer* Zeile unterbringen. Wir trennen sie durch ein Semikolon(;). Wir wollen also das Programm, das wir auf der letzten Seite in zwei

Zeilen geschrieben haben, in einer Zeile unterbringen – z. B., um Platz zu sparen. Die neue Zeile 10 sieht dann so aus:

```
10 PRINT "3*7=";3*7
```

Und nachdem wir das Programm gestartet haben, kommt heraus:

```
10 PRINT "3*7=";3*7
RUN
3*7=  21

READY.
```

(Was würde passieren, wenn man statt des Semikolon (;) ein Komma (,) verwenden würde? Probieren wir's:

```
10 PRINT "3*7=",3*7
RUN
3*7=          21
```

Aha, durch das Komma entsteht ein Zwischenraum. Genau gesagt, das Komma unterteilt den Bildschirm in vier Spalten. Das läßt sich durch folgendes Mini-Programm zeigen:

```
10 PRINT "*","*","O","O"
20 PRINT
30 PRINT "*","*","*","*"
40 PRINT
50 GOTO 10
```

Nach dem Starten des Programms siehst du einen Computerfilm ablaufen. Die Zeichenketten laufen genau am Anfang von vier Spalten über den Bildschirm. Es sind jene Spaltenanfänge, die entstehen, wenn wir die vier verschiedenen Zeichen nach einem PRINT-Befehl durch ein Komma trennen. Die Zeichen in den vier Spaltenanfängen laufen ziemlich schnell über den Bildschirm. Wenn man eine Zeitlang fest hinschaut, gibt es wieder einen merkwürdigen optischen Effekt:

Man kann nicht mehr genau sagen, ob die Zeichen von unten nach oben laufen oder umgekehrt. Auch wie schnell das geht, läßt sich bald kaum noch unterscheiden, wenn man von einem dieser „Perlenfäden“ auf den anderen schaut.

Tipp doch mal, während das Programm läuft, auf die etwas größere Taste links (mit den Buchstaben CTRL), und bleibe mit dem Finger auf dieser CTRL-Taste. Was passiert? Die Zeilen folgen nun sehr viel langsamer aufeinander. Es sieht fast so aus, als würde nun unser Computerfilm in Zeitlupe ablaufen.

Die Bewegung der Zeilen, während unser Programm läuft, nennt man Scrolling. Scrolling ist ein englisches Wort und bedeutet rollen oder abrollen. (Man kann sich ja wirklich ganz gut vorstellen, daß da wie auf einer Rolle etwas weiterbewegt wird.) Durch das Drücken der CTRL-Taste kann man also dieses schnelle Scrolling langsamer machen.

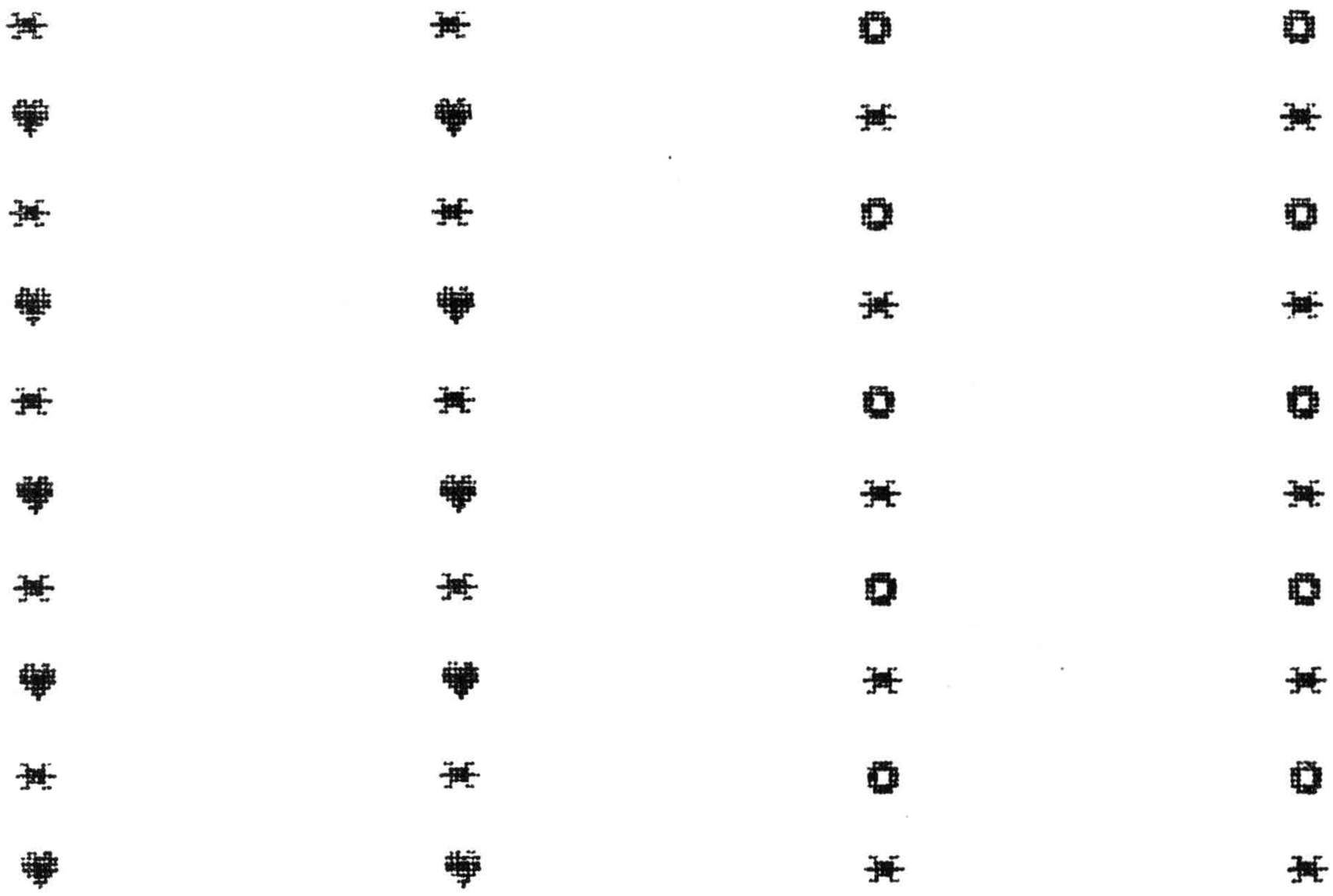
Das kann zum Beispiel besonders nützlich bei langen Listings sein. Die CTRL-Taste verfügt auch über andere Möglichkeiten. Doch darüber will ich erst später erzählen. Wenn man die CTRL-Taste wieder losläßt, laufen die Zeilen wieder mit der vorigen, höheren Geschwindigkeit.

Jetzt stoppen wir aber unser Programm, bevor es uns die Augen verdreht. Mit welcher Taste? (Richtig, RUN/STOP!) Auf dem Bildschirm sieht es dann ungefähr so aus wie auf der Seite 35 oben.

BREAK (gesprochen „brejk“) bedeutet im Englischen soviel wie Unterbrechung oder Pause.

BREAK IN 30

heißt: Der Computer hat bei deinem Tastendruck



```
BREAK IM 30
READY.
```

das Programm an der Programmzeile 40 unterbrochen. Das hängt davon ab, wann du die Taste gedrückt hast. Der Computer hält das Programm dort an, wo er gerade beim „Abarbeiten“ angelangt ist.

Bei mehreren Unterbrechungen kann das Programm auch an verschiedenen anderen Zeilen angehalten werden. Das kannst du ausprobieren, wenn du das Programm mit

RUN

wieder startest und mit der RUN/STOP-Taste noch einmal unterbrichst. Übrigens gibt es noch ein zwei-

tes Befehlswort in BASIC, das das Programm nach einer Unterbrechung wieder zum Laufen bringt. Es heißt

CONT

und ist die Abkürzung für das englische Wort „continue“ (gesprochen „kontinjuh“; das heißt auf deutsch „fortsetzen“ oder „weitermachen“). Der Unterschied zu RUN?

- Mit RUN sagen wir dem Computer, er soll das Programm wieder von vorne anfangen.
- Mit CONT sagen wir ihm, er soll da mit dem Programm weitermachen, wo er unterbrochen wurde.

Hier ist ein kleines Programm zum Rechnen-Üben. Damit kannst du dir durch die RUN/STOP-Taste eine Rechnung herausholen. Wenn die Rechnung gelöst ist, kann man den Computer wieder als „Quizmaster“ einsetzen.

Das Programm könnte dann zum Beispiel so aussehen wie auf der folgenden Seite oben.
Du kannst auch andere Zahlen, leichtere oder schwierigere Rechnungen einsetzen. Das weiße Herzchen auf schwarzem Grund löscht vor jeder neuen Aufgabe den Bildschirm. Innerhalb der Anführungszeichen erhalten wir es durch

SHIFT und CLR/HOME

```
10  PRINT"[CLR][25 DOWN]  -";:4
20  PRINT"[3 DOWN]  +";:4
30  PRINT"[9 DOWN]  *";:4
40  PRINT"[8 DOWN]  +";:4
50  PRINT"[13 DOWN]  -";:4
60  PRINT"[36 DOWN]  /";:4
70  PRINT"[6 DOWN]  *";:4
80  PRINT"[16 DOWN]  /";:4
90  PRINT"[29 DOWN]  -";:4
100  PRINT"[79 DOWN]  +";:4
110  PRINT"[5 DOWN]  *";:4
120  PRINT"[8 DOWN]  /";:4
130  GOTO 30

READY.
```

Das [CLR] macht unseren Bildschirm „sauber"!

Unser kleines Rechenübungsprogramm

Sobald du dein Programm nun startest, rasen in der obersten Bildschirmzeile die Aufgaben durch. Mit einem Druck auf die RUN/STOP-Taste unterbrichst du das Programm, und *eine* Aufgabe bleibt stehen. Die sollst du nun ausrechnen.

Um herauszufinden, ob du richtig gerechnet hast, kannst du den Computer im „Direktmodus" fragen. Das heißt, du schreibst keine Zeilennummer vor deinen Befehl.

Im Direktmodus schreibt man also kein Programm. Das hat hier den Vorteil, daß unser vorheriges Programm nicht gestört wird. Falls wir dem Computer zwischendurch im Direktmodus ein paar Fragen gestellt haben, ist unser Programm anschließend noch genauso vorhanden wie vorher. (Kann

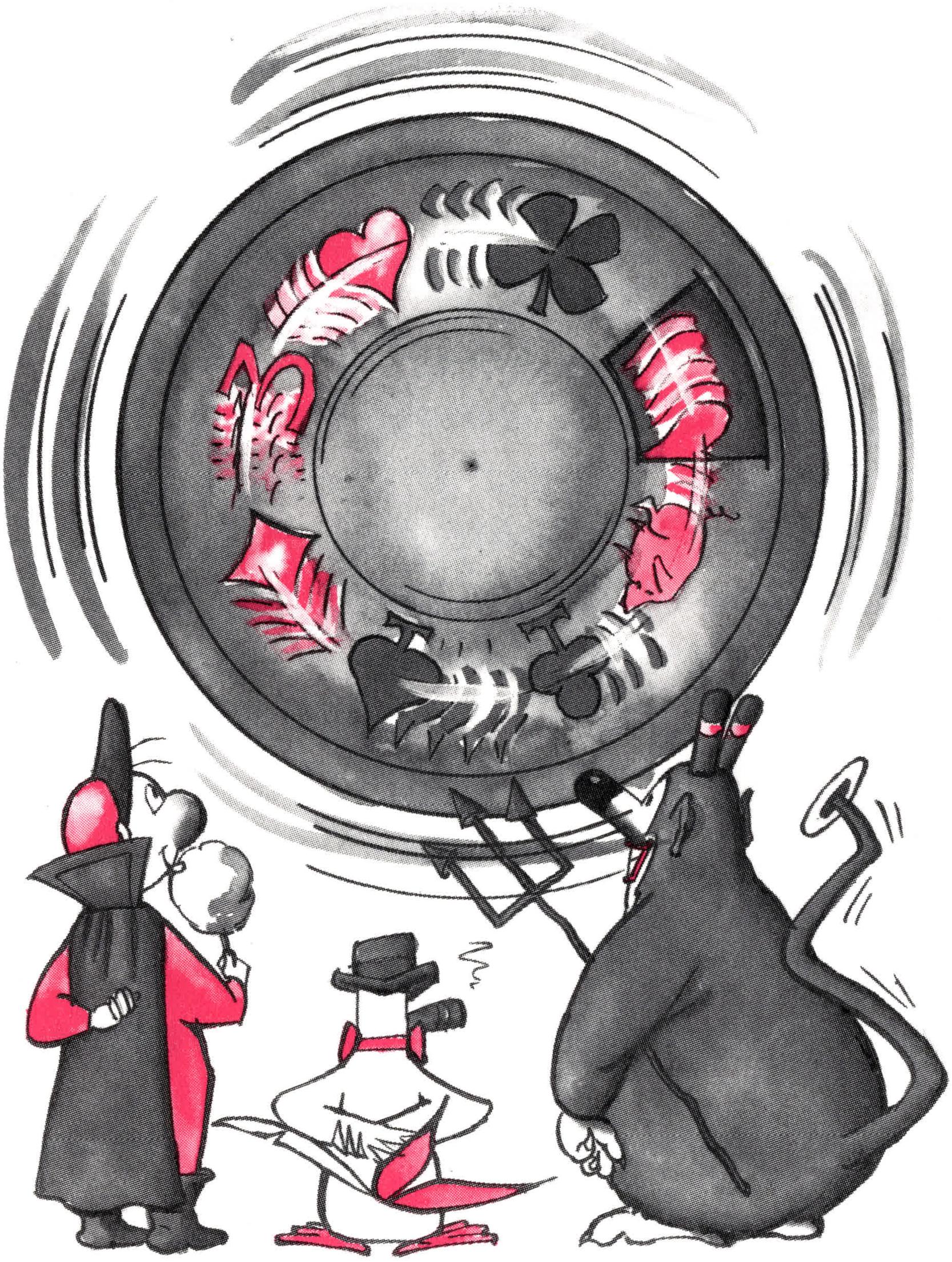
man ja durch LIST immer wieder nachprüfen.)

Soll der Computer wieder den Quizmaster spielen, dann behaupten wir im Direktmodus mit PRINT eine Lösung für die jeweilige Rechnung. Und durch 0 *(falsch!)* oder –1 *(richtig!)* gibt er die Antwort. So kann das aussehen:

```
29  -  4

BREAK  IN  110
READY.
PRINT29-4=28
 0

READY.
PRINT29-4=25
-1

READY.
```

Mit diesem Programm kann man nun ein bißchen herumspielen. Du kannst es zum Beispiel immer länger oder die Aufgaben immer schwieriger machen. Du kannst es immer wieder ändern. Man kann auch ein bißchen Text einbauen, zum Beispiel die Namen der Mitspieler oder eine Aufforderung wie: „Rechne:“

Weil du mittlerweile so viele Möglichkeiten kennst, mit deinen Computerprogrammen zu arbeiten, zu radieren und zu verbessern, kannst du ruhig auch etwas ganz Neues oder sogar etwas „Verrücktes“ ausprobieren: Vielleicht bekommst du auf diese Weise eine Art „Glücksspiel“ zustande. Oder ein nettes



kleines Quatsch-Spiel mit verschiedenen Aufgaben
in den PRINT-Zeilen wie

GEHE DREIMAL RÜCKWÄRTS IM KREIS HERUM
SINGE EIN LIED VOR

und so weiter.

Mit einem weiteren BASIC-Wort können wir sogar die Lösungen im Programm verstecken.
Es heißt

REM

und ist die Abkürzung von „Remark“, dem englischen Wort für Anmerkung, Notiz. Was hinter einem REM-Befehl steht, hat keine Bedeutung für den Programmablauf. Es ist nur eine Anmerkung für den Programmierer. Zum Beispiel kann man in REM-Zeilen seinen Namen unterbringen und auch ein bißchen Ornamente zeichnen. So etwa:

```
10 REM "------------------------------------------------------------"
20 REM "DIESES SUPERPROGRAMM HAT"
30 REM "          GESCHRIEBEN:                                      "
40 REM "♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
50 REM "                                                            "
60 REM "------------------------------------------------------------"
70 REM
```

Hier steht dein Name.

Wie gesagt: Diese Zeilen haben keinen Einfluß auf das Programm. Der Computer tut so, als wären sie nicht da. Nur wenn es ans LISTen des Programms geht, tauchen sie wieder auf (siehe Seite 41 oben!).

Ob das wirklich stimmt, kannst du selber ausprobieren, mit den beiden Anweisungen

```
10 REM KLEINES RECHENSPIEL
20 REM DIE LETZTE ZAHL DARF GEAENDERT
   WERDEN
30 PRINT"{CLR}{25 RIGHT} -";4
40 PRINT"{3 RIGHT} +";4
50 PRINT"{9 RIGHT} *";4
60 PRINT"{8 RIGHT} +";4
70 PRINT"{13 RIGHT} -";4
80 PRINT"{36 RIGHT} /";4
90 PRINT"{6 RIGHT} *";4
100 PRINT"{16 RIGHT} /";4
110 PRINT"{29 RIGHT} -";4
120 PRINT"{79 RIGHT} +";4
130 PRINT"{5 RIGHT} *";4
140 PRINT"{8 RIGHT} /";4
150 GOTO30
```

Hier sind 2 REM-Zeilen im Programm eingebaut.

RUN

und

LIST

Ebenso wie sich „Nachrichten“ durch REM-Zeilen in einem Programm verstecken lassen, kann man auch Lösungen von Aufgaben darin verstecken, die durch

LIST

herausgeholt werden können.

Der Max hat ein Programm geschrieben, mit dem er sich „abfragen“ läßt. Und die Lösungen hat er in den REM-Zeilen versteckt.

```
10 REM KLEINES RECHENSPIEL
20 REM DIE LETZTE ZAHL DARF GEAENDERT
    WERDEN
30 PRINT"{CLR}{25 RIGHT} -";4
35 REM 21
40 PRINT"{3 RIGHT} +";4
45 REM 7
50 PRINT"{9 RIGHT} *";4
55 REM 36
60 PRINT"{8 RIGHT} +";4
65 REM 12
70 PRINT"{13 RIGHT} -";4
75 REM 9
80 PRINT"{36 RIGHT} /";4
85 REM 9
90 PRINT"{6 RIGHT} *";4
95 REM 24
100 PRINT"{16 RIGHT} /";4
105 REM 4
110 PRINT"{29 RIGHT} -";4
115 REM 25
120 PRINT"{79 RIGHT} +";4
125 REM 83
130 PRINT"{5 RIGHT} *";4
135 REM 20
140 PRINT"{8 RIGHT} /";4
145 REM 2
150 GOTO30
```

REM-Zeilen werden ziemlich wichtig, wenn die Programme länger werden. Man kann sich dann ins Listing schreiben, welche Aufgabe zum Beispiel die einzelnen Programm-Teile erfüllen sollen. Mein Vorschlag: Ab jetzt wollen wir in unsere längeren Programme immer die eine oder andere REM-Zeile einfügen. In ihr steht, wie das Programm heißt, was es machen soll oder welcher Teil unserer Aufforderungen an den Computer nun dran ist.

Natürlich kannst du auch Lösungen für Rechnungen in REM-Zeilen verstecken. Und daraus kann man wieder ein Rechen-Spiel machen. Die Rechnung wird, wie in unserem Programm von vorhin, mit Hilfe der RUN/STOP-Taste ausgewählt. Und durch das

```
LIST
```

kannst du die REM-Zeilen auf den Bildschirm holen und nachsehen, ob du richtig gerechnet hast.

```
6 * 4

BREAK IN 90
READY.
24
LIST95

95 REM 24

READY.
```

Anhand solcher einfacher Computer-Programme lassen sich schon ganz hübsche Rechen-Übungen

zusammenstellen. Eine Rechen-Übung ist auch unser erstes großes Spiel. Doch keine Angst, bei unserem Rechenmampfer geht es spannend und lustig zu. Die Anleitung zum Spiel findest du auf diesen beiden Seiten.

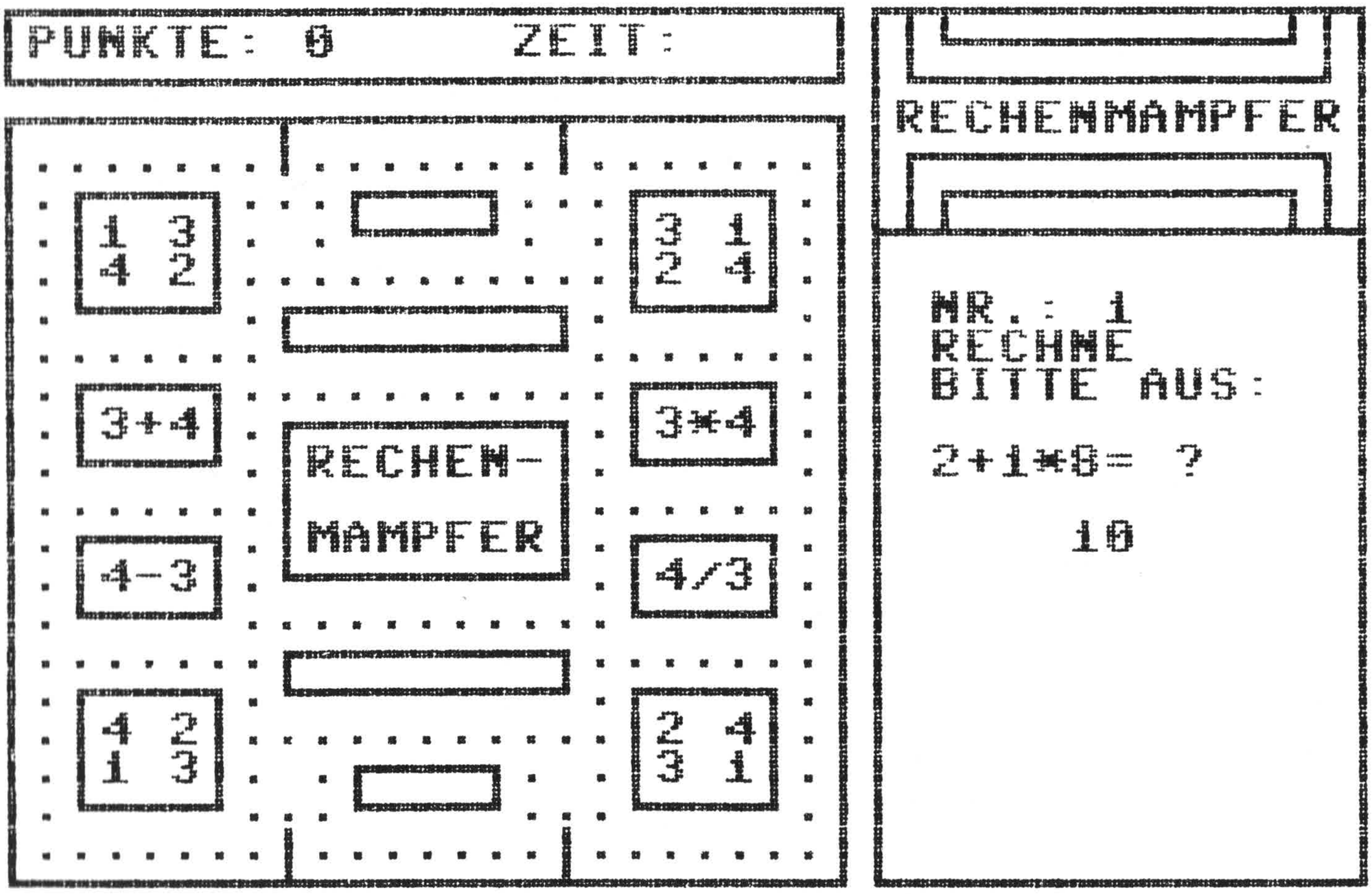
RECHENMAMPFER

In diesem Spiel geht es darum, 10 Rechenaufgaben zu lösen. Für jede richtige Lösung darf man 30 Sekunden lang den Rechenmampfer durch sein Labyrinth schicken. Dabei soll er so viele von den Punkten mampfen wie möglich. Sie werden oben am Bildschirm vom Computer gezählt. Für jede falsche Lösung werden dem Spieler aber Punkte abgezogen. Danach kann er die Aufgabe noch einmal rechnen.

Wenn du deine Lösung für richtig hältst, bestätige sie mit RETURN. Mit der Leertaste kannst du sie zum Verschwinden bringen und eine andere Lösung eingeben – solange du noch nicht RETURN gedrückt hast.

Der Rechenmampfer wird mit den folgenden Buchstaben-Tasten gesteuert:

I	der Rechenmampfer geht nach oben



In diesem Irrgarten mußt du möglichst viele Punkte mampfen

J der Rechenmampfer geht nach links

K der Rechenmampfer geht nach rechts

M der Rechenmampfer geht nach unten

Der Rechenmampfer behält seine Richtung immer so lange bei, bis er an eine Linie stößt oder eine andere Taste gedrückt wird.

Mit RUN STOP + RESTORE kannst du das Programm stoppen.

Neu anfangen: LOAD "S", 8, 1

Der Super-Rechner und das Geheimnis der „Stellvertreter“

Falls in einer Programmzeile oder hinter einem BASIC-Befehl mehrere Dinge stehen, die der Computer ausführen oder auf den Bildschirm bringen soll, dann müssen sie durch ein Zeichen getrennt werden.

Das ist eine heikle Angelegenheit. Denn damit nimmt es der Computer sehr genau. Vergißt man so ein Zeichen, kann es einem leicht das ganze Programm durcheinanderbringen. Nicht anders ist es, falls wir ein *falsches* Zeichen eingegeben haben. Der Computer kann ja nicht wissen, was wir „meinen“. Er weiß nur genau das, was wir in seiner Sprache geschrieben haben. Nicht mehr und nicht weniger.

Überdies ist der Gebrauch dieser Zeichen von Computer zu Computer sehr unterschiedlich. Ein Komma kann unter Umständen bei unserem Commodore 64 etwas anderes bedeuten als zum Beispiel auf einem CPC 464 oder einem Sinclair.

Beim Abtippen von Listings mußt du sehr genau auf die folgenden Zeichen achten:

;	,	.	:

Hat man in einem normalen geschriebenen Text einmal ein Komma oder ein Semikolon vergessen oder falsch gesetzt, ist das vielleicht nicht besonders schön. Aber dennoch weiß einer, der den Text liest, meistens doch recht genau, was damit gemeint ist. In

einem BASIC-Programm ist das anders. Da kann es vorkommen, daß der Computer durch ein falsches Zeichen auf einmal die Welt nicht mehr versteht. Oder er macht etwas ganz anderes, als du eigentlich von ihm wolltest.

Probieren wir noch ein bißchen mit diesen Zeichen herum.

Das machen wir für den Augenblick am besten im Direktmodus. Die einfachste Lösung, zwei Rechnungen zu trennen, ist ein Semikolon:

```
PRINT3*4;4*4
 12   16

READY.
```

RETURN

Mit Hilfe des Kommas schaffen wir einen Zwischenraum. Der Bildschirm wird in vier Spalten geteilt; zwei davon sind jetzt belegt:

```
PRINT3*4,4*4
 12            16

READY.
```

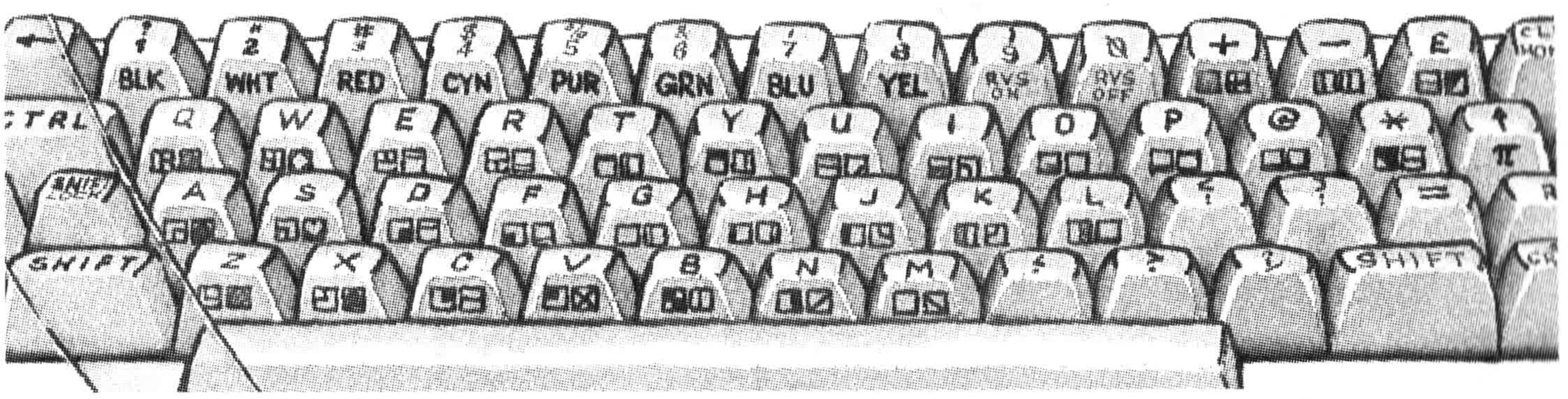
Vielleicht kannst du dir das noch einmal verdeutlichen. Dazu machst du dir am besten (wie im ersten Buch beschrieben) wieder eine „Kästchen-Version“ eines Bildschirms. Und da trägst du dann die vier Spalten ein. Das sieht so aus:

Diese Spalten, die wir durch das Komma erhalten,

können wir übrigens auch ausnützen, falls wir bei einer Computer-Zeichnung oder einem Computer-Film nicht so viele Leerzeichen verwenden wollen.

Ein Beispiel ist unser „Verkehrs-Film". Hier ist das Programm dazu auf Seite 49.

So ein Programm kann man abtippen, wenn mal an einem Nachmittag schlechtes Wetter ist. Noch lehrreicher ist es für dich allerdings, du denkst dir für die Autos und Flugzeuge eigene Zeichenkombina-



tionen aus. Dazu verwendet man die Grafikzeichen, die auf den Tasten an der Vorderseite aufgemalt sind.

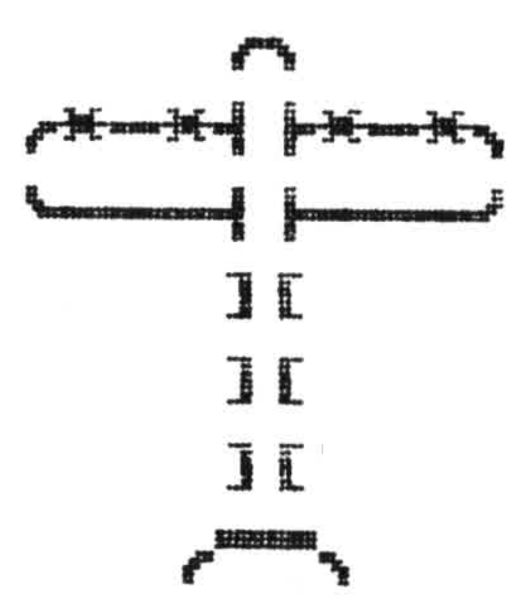
Das rechte Grafik-Zeichen erhalten wir zusammen mit

SHIFT

Das linke Grafikzeichen erhalten wir zusammen mit der Taste

C=

```
100 PRINT,"|      .          |"
110 PRINT,"|                 |"
120 PRINT,"|      .          |"
130 PRINT,"|           ┌──┐  |"
140 PRINT,"|      .    ├──┤  |"
150 PRINT,"|           ├──┤  |"
160 PRINT,"|      .    |  |  |"
170 PRINT,"|           └──┘  |"
180 PRINT,"|      .          |"
190 PRINT,"|                 |"
200 PRINT,"|      .          |"
210 PRINT,"|                 |"
220 PRINT,"|      .          |"
230 PRINT,"|                 |"
240 PRINT,"|      .          |"
250 PRINT,"|                 |"
260 PRINT,"|      .          |"
270 PRINT,"|                 |"
280 PRINT,"|      .          |"
290 PRINT,"|                 |"
300 PRINT,"|      .          |"
310 PRINT,"|                 |"
320 PRINT,"|      .          |"
330 PRINT,"|                 |"
340 PRINT,"|      .          |"
350 PRINT,"|                 |"
360 PRINT,"|      .          |
370 PRINT,"|                 |
380 PRINT,"|      .          |
390 PRINT,"|                 |
400 PRINT,"|      .          |
410 PRINT,"|                 |
420 PRINT,"|      .          |
430 PRINT,"|                 |"
440 PRINT,"|      .          |"
450 PRINT,"|                 |"
460 PRINT,"|      .          |"
470 PRINT,"|                 |"
480 PRINT,"|      .     ⌒    |"
490 PRINT,"|           oo    |"
500 PRINT,"|      .    ▪ ▪   |"
510 PRINT,"|           ▒▒    |"
520 PRINT,"|      .    ▒▒    |"
530 PRINT,"|           ▒▒    |"
540 PRINT,"|      .     |    |"
550 PRINT,"|           ▒▒    |"
560 PRINT,"|      .    ▒▒    |"
570 PRINT,"|           ▒▒    |"
580 PRINT,"|      .    ╥╥    |"
590 PRINT,"|                 |"
600 GOTO 100
```



Dies ist unser Verkehrsfilm-Programm

7 9091-20

So sehen die Grafikzeichen aus, die zur Q-Taste gehören:

```
  Q-TASTE
MIT SHIFT  :  ●

MIT C=     :  ▌
```

Bis sich da die geeigneten Zeichen zu einem richtigen Bild zusammensetzen, muß man ein bißchen herumprobieren. Aber es macht auch Spaß.

Indem wir das Komma benützen, haben wir in unserem Beispiel die Straße gleich von der zweiten Spalte aus angefangen. Da konnten wir demnach die zehn Leerzeichen innerhalb der Anführungszeichen sparen. Das macht unsere Programmzeilen kürzer, und schon im Programm läßt sich erkennen, was nachher auf dem Bildschirm zu sehen ist.

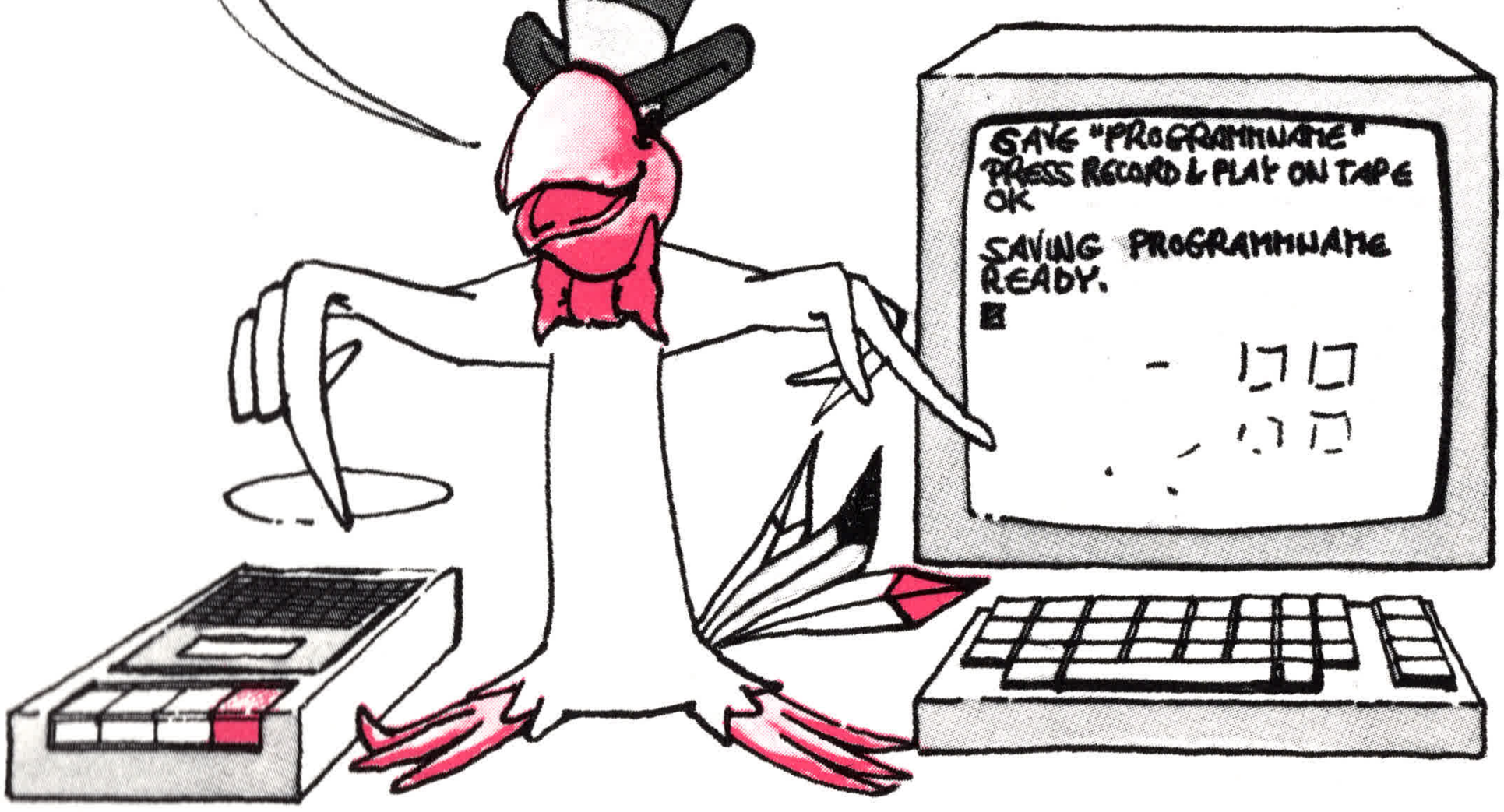
Hast du dein eigenes Auto- und Flugzeug-Programm geschrieben? Wäre bei soviel Mühe ziemlich schade, wenn es beim Computerabschalten verlorengeht. Da gibt's nur eines: **Abspeichern,** also auf einen Datenträger wie Kassette oder Diskette bringen.

Hier noch einmal alles, was du beim Abspeichern oder Laden beachten mußt:

ABSPEICHERN AUF KASSETTE:

– Vor dem Abspeichern prüfen, ob der Computer das Programm wirklich im Gedächtnis hat — also: LIST eintippen! Dann...

... – unbespielte Kassette einlegen
– an den Anfang zurückspulen
– eintippen: SAVE "PROGRAMMNAME"
– Meldung vom Computer:
PRESS RECORD & PLAY ON TAPE
– Der Rekordet läuft; der Computer meldet: OK
SAVING PROGRAMMNAME
READY.



LADEN VON DER KASSETTE:

- KASSETTE an den Anfang zurückspulen
- LOAD "PROGRAMMNAME" eintippen
- Computer meldet:
  PRESS PLAY ON TAPE
- PLAY-Taste auf dem Rekorder drücken
- Computer meldet:
  OK
  und kurz darauf:
  SEARCHING FOR PROGRAMMNAME
  FOUND PROGRAMMNAME
  LOADING
  READY
- mit LIST überprüfen, ob gewünschtes Programm wirklich im Computergedächtnis

ABSPEICHERN AUF DISKETTE:

- formatierte* (und initialisierte*) Diskette in das Diskettenlaufwerk einlegen.
  (*diese beiden Begriffe erklärt das Handbuch)
- eintippen: SAVE "PROGRAMMNAME", &
- Die Diskette dreht sich.
- Während des Abspeicherns leuchtet das rote Lämpchen.
- Der Computer meldet:
  SAVING PROGRAMMNAME
  READY.
- Nachschauen, ob Programm auch wirklich abgespeichert – also folgendes eintippen:
  LOAD "$", &
- Der Computer meldet:
  SEARCHING FOR $
  LOADING
  READY.
- LIST eintippen – der Computer zeigt sein Inhaltsverzeichnis:
  1 "PROGRAMMNAME" PRG

das ist jetzt anders!

Jetzt aber zurück zu unseren Trennungszeichen im BASIC-Programm. Nach dem Semikolon [;] und dem Komma [,] können wir jetzt mal den Punkt [.] ausprobieren.

Geben wir im Direktmodus zum Beispiel folgendes ein:

```
PRINT 2+3.5–3
```

Dann kommt nach dem Drücken der RETURN-Taste heraus:

```
PRINT2+3.5-3
 2.5

READY.
```

Was bedeutet wohl dieses Ergebnis? Ganz einfach: Der Punkt ist für den Computer *kein* Trennungszeichen. Für ihn bedeutet er das, was für uns beim Rechnen normalerweise das Komma ist. Er rechnet also zuerst 2 plus 3,5 (also dreieinhalb) und zieht dann davon 3 ab. Und das gibt ja wirklich 2,5 (also zweieinhalb). Das kannst du nun ein paarmal ausprobieren. Du gibst dem Computer dazu Rechnungen auf, die „nicht aufgehen".
Zum Beispiel

```
PRINT 3/2
```

Nach dem Drücken der RETURN-Taste antwortet der Computer mit der Lösung: 1.5

Übrigens: Auch die Taschenrechner verwenden statt des Kommas bei den nicht-ganzen Zahlen einen Punkt. Man gewöhnt sich sehr schnell daran.

Nun können wir ja noch ein weiteres Zeichen ausprobieren. Wie wäre es mit einem Doppelpunkt [:]?

Stellen wir also dem Computer die folgende Aufgabe:

```
PRINT"3*7":3*7
```

Das scheint er gar nicht zu mögen. Er meldet uns, daß wir einen Fehler in der BASIC-Sprache gemacht haben und er uns nicht versteht. Das sieht bei ihm so aus:

```
PRINT"3*7":3*7
3*7
?SYNTAX  ERROR
READY.
```

Den ersten Teil der Aufgabe löst er ganz richtig. Er bringt auf den Bildschirm, was zwischen den Anführungszeichen steht. Aber den zweiten Teil nach dem Doppelpunkt, nämlich auch die Lösung auszugeben, hat er offenbar nicht kapiert.

Der Doppelpunkt scheint im BASIC etwas ganz anderes zu bedeuten als in der normalen Schrift. Er trennt nämlich eine Hälfte von der anderen Hälfte einer Programmzeile ab. Man könnte sich vorstellen, der Doppelpunkt ist wie eine Mauer zwischen zwei Aufgaben in einer Zeile.

Das heißt aber auch: hinter dem Doppelpunkt beginnt etwas Neues. Und deshalb muß dem Computer noch einmal gesagt werden, was er tun soll. Wir müssen folglich noch einmal ein „Befehlswort“ verwenden. Und da wir eine Lösung erwarten, ist das natürlich unser

PRINT

Jetzt können wir also dem Computer die Aufgabe noch mal stellen, und zwar so:

```
PRINT"3*7=":PRINT3*7
3*7=
 21

READY.
```

Fällt dir dabei etwas auf? Genau, die Lösung steht wieder in einer neuen Zeile. Der Doppelpunkt bewirkt, daß in einer einzigen Zeile mehrere Befehle ausgeführt werden, ganz so, als seien es zwei Zeilen. Und der Computer beginnt jedesmal mit einer neuen Zeile.

Da haben wir schon einen der Tricks, wie man ein Programm kürzer und damit schneller machen kann: Man schreibt mehrere Befehle in eine Zeile, sofern sie sehr einfach sind oder – das kommt bald auch bei unseren Programmen vor – falls sie etwas miteinander zu tun haben.

Nehmen wir als Beispiel dieses kleine Programm:

```
10 PRINT,"*","    *"
20 PRINT
30 PRINT,"    *","*"
40 PRINT
50 GOTO10
```

Weißt du noch, was ein PRINT allein bedeutet? Klar: Eine Zeile wird leer gelassen.

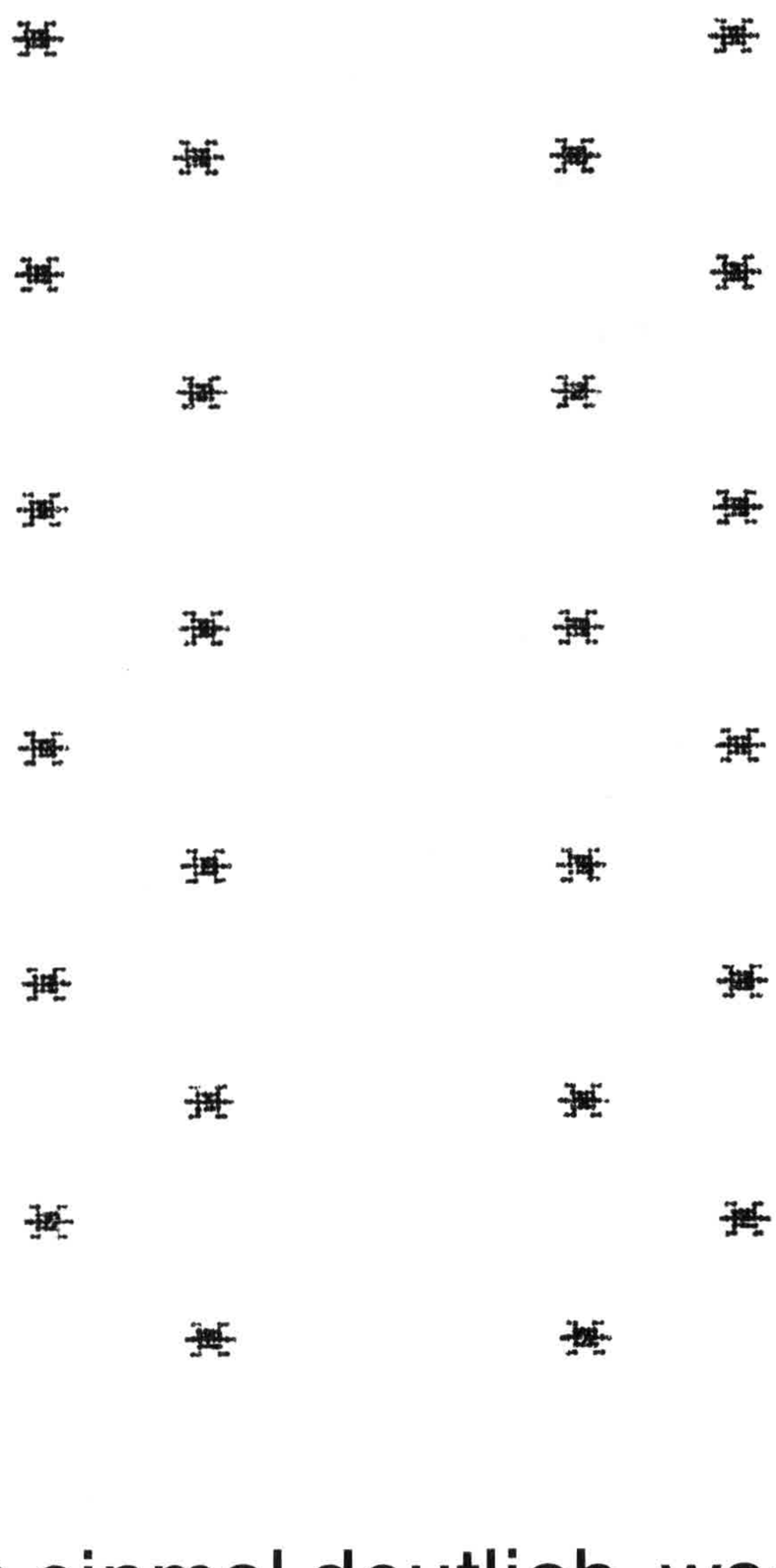
Das Komma setzt den Beginn auf die nächste der vier Bildschirm-Spalten.

Mit GOTO 10 schicken wir den Computer wieder an den Anfang des Programms.
Sonst könnte das Programm ewig weiter laufen, es sei denn, man unterbricht es mit RUN/STOP.

Dieses Programm könnten wir in einer einzigen Programmzeile unterbringen, nämlich so (lösche aber bitte vorher das Gedächtnis deines Computers):

```
10 PRINT,"*","    *":PRINT:PRINT,"    *","*"
:PRINT:GOTO10
```

Dieses Programm, das aus einer langen Zeile besteht, läuft genauso ab wie jenes von vorhin, das aus 5 Zeilen besteht. Nämlich so:



Daran wird noch einmal deutlich, was uns schon bei den Computer-Bildern begegnet ist: Eine Zeile im Programm darf durchaus länger sein als eine Zeile auf dem Bildschirm.

```
10 PRINT,"*","    *":PRINT:PRINT,"    *","*":PRINT:GOTO10

READY.
```

Man darf allerdings nicht zu viele Befehle in eine Zeile packen. Erstens wird das Programm dann schnell ziemlich unübersichtlich. Stell dir einmal vor, du müßtest mittendrin etwas ändern! Und zweitens verkraftet der Computer auch keine beliebig großen Mengen an Informationen in einer Programmzeile.

Wollen wir testen, wieviel der Computer „verträgt“? Dazu erweitern wir ein Programm in einer Zeile immer wieder um eine Sternchen- und eine Leerzeile und lassen es jedesmal von neuem laufen. Mal sehen, wie weit es geht. Also:

```
10PRINT"*":PRINT:PRINT"**":PRINT:PRINT"*
**":PRINT
```

und so weiter. Aber jedesmal das Programm neu starten, damit man merkt, wann der Computer nicht mehr mag.
So geht es noch:

```
10PRINT"*":PRINT:PRINT"**":PRINT:PRINT"*
**":PRINT:PRINT"****":PRINT:PRINT"*****"
:PRINT
```

Und so geht es nicht mehr:

```
10 PRINT"*":PRINT:PRINT"**":PRINT:PRINT"
***":PRINT:PRINT"****":PRINT:PRINT"*****
":PRINT:PRINT"******":PRINT

?SYNTAX  ERROR
READY.
```

Eine Programmzeile soll somit auf keinen Fall länger sein als zwei Bildschirmzeilen. Das gilt nicht nur für die verschiedenen Befehle, die wir unserem Computer geben, sondern genauso für Texte, die wir ihn auf dem Bildschirm schreiben lassen. Du kannst ja ausprobieren, wieviel Buchstaben er hinter einem PRINT in einer Zeile verträgt.

Na, wie viele sind es?

Hier wird es ihm zuviel:

```
10 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ*ABC
DEFGHIJKLMNOPQRSTUVWXYZ*ABCDEFGHIJKLMNOP
Q

?SYNTAX  ERROR
READY.
```

So erfährt man immer wieder durch Ausprobieren, was der Computer kann und was er nicht kann. Was dem Computer besonders gut liegt: Na klar – Rechnen! Das macht er mindestens genauso gut und schnell wie ein Taschenrechner. Aber er kann noch viel mehr. Statt mit einer festen Zahl kann der Computer auch mit den sogenannten Variablen rechnen.

Variablen sind „Stellvertreter“ für Zahlen, denen man ganz verschiedene Werte zuweisen kann. Wir schreiben sie als Buchstaben. Eine Rechnung mit einer Variablen könnte so aussehen:

```
10 PRINT A*3
```

Das ist natürlich noch kein vernünftiges Programm. Denn A mal 3 gibt eigentlich keinen rechten Sinn.

Jedenfalls nicht, solange keiner weiß, was mit A eigentlich gemeint ist. Es muß erst klar sein, für welche Zahl das „A“ als Stellvertreter steht. Besser ist es, wir sagen dem Computer in der ersten Zeile gleich, was für eine Zahl wir als A haben wollen:

```
10 A=4
```

Das bedeutet: In unserem Programm steht der Buchstabe A als Stellvertreter für die Zahl 4. In die zweite Zeile können wir jetzt die Rechnung mit dem Stellvertreter schreiben. Starten wir das fertige Programm!

```
10 A=4
20 PRINTA*3
RUN
 12
```

Die Lösung ist richtig, denn 4 mal 3 ist ja wirklich 12. Jetzt ändern wir die erste Zeile, zum Beispiel in

```
10 A=5
```

Das Programm ergibt als Lösung:

```
10 A=5
20 PRINTA*3
RUN
 15

READY.
```

Stimmt ebenfalls, denn 5 mal 3 ist 15.

Das Gleichheitszeichen (=) hat für den Computer eine etwas andere Bedeutung als beim für uns üblichen Rechnen – jedenfalls wenn es zusammen mit einem Stellvertreter verwendet wird. Es besagt nicht, daß A ein für allemal genau gleich 3 ist. Da hätten wir ja unserer Zahl 3 nur einen neuen Namen gegeben. Es bedeutet vielmehr daß A *jetzt* 3 *sein soll,* und zwar nur so lange, bis dem Computer etwas anderes gesagt wird. Das sieht man ganz deutlich, wenn wir aus den beiden Programmen von vorhin ein einziges machen:

```
10 A=4
20 PRINTA*3
30 A=5
40 PRINTA*3
```

In Zeile 10 sagen wir dem Computer: A soll der Stellvertreter für die Zahl 4 sein.
In Zeile 20 sagen wir ihm: Multipliziere diese Zahl mit 3.
In Zeile 30 sagen wir ihm: Ab jetzt soll A nicht mehr der Stellvertreter für die Zahl 4, sondern für die Zahl 5 sein.
In Zeile 40 sagen wir ihm: Multipliziere diese (neue) Zahl mit 3.

Laß das Programm laufen; wir erhalten zwei Lösungen, da wir ja dem Stellvertreter A zwei verschiedene

Zahlen zugewiesen haben.

```
10 A=4
20 PRINTA*3
30 A=5
40 PRINTA*3
RUN
 12
 15

READY.
```

In der Zeile 30 wurde dem Computer eigentlich gesagt, daß die Zahl für den Stellvertreter A um eins höher ist als vorher. Das könnte man auch anders schreiben, als wir es getan haben! Nämlich so:

```
30 A=A+1
```

Das wäre natürlich völliger Unsinn, wenn das Gleichheitszeichen des Computers dasselbe bedeuten würde, wie wir es vom Matheunterricht her kennen. Aber hier bedeutet es: Von nun an soll die Zahl für den Stellvertreter A um 1 höher sein als vorher. Laß das neue Programm laufen. Kommt wirklich dasselbe Ergebnis heraus?

Spielen wir noch ein bißchen mit diesem Programm herum. Was wäre zum Beispiel, wenn wir vergessen hätten, dem Stellvertreter A eine Zahl zuzuweisen? Um das herauszufinden, mußt du lediglich die Zeile 10 löschen und dann das Programm starten.

```
20 PRINTA*3
30 A=A+1
40 PRINTA*3
RUN
 0
 3

READY.
```

Zuerst sehen die beiden Ergebnisse, die der Computer „ausspuckt“, etwas seltsam aus. Aber vielleicht kommst du mit ein wenig Nachdenken darauf, welche Zahl der Computer dem Stellvertreter A zugewiesen hat, nachdem wir ihm ja keine angegeben haben. Na?

Aha: Ein Stellvertreter, dem du als Programmierer keine Zahl zugeordnet hast, bekommt vom Computer automatisch die 0 zugewiesen. Das neue A in Zeile 30 bekommt dann den Wert 0+1, und das ergibt 1.

Daß das wirklich so ist, kannst du im Direktmodus nachprüfen. Was ergibt zum Beispiel:

```
PRINTA+4
```

Bekommt dein Computer eine 4 oder eine 5 heraus? Falls es eine 5 ist, hast du irgendwas vergessen. Kommst du drauf? Vielleicht versuchst du es einmal mit

```
LIST
```

O.K., jetzt löschen wir das Gedächtnis des Computers wieder, damit er nicht treu und brav immer wieder die Zeile 30 ausführt. Und nun kannst du ein paar Experimente mit dem Stellvertreter machen.

Dabei müssen die Rechnungen nicht immer mit dem Stellvertreter anfangen. Er kann auch in einer Rechenkette stehen. Und du kannst auch im Direktmodus dem Stellvertreter A irgendeine Zahl zuweisen. Die gilt dann so lange, bis du eine neue wählst. Alle diese Rechenarten sind möglich: Dabei kann man auch eine zweite und dritte Variable (Stellvertreter) einsetzen, die andere Buchstaben als Namen haben.

Beim Herumprobieren mit Zahlen und Variablen, den Stellvertretern, stößt du wahrscheinlich ab und zu auf Ergebnisse, die ihr im Mathematik-Unterricht noch nicht besprochen habt. Hat der Stellvertreter in unserem Rechenbeispiel gerade den Wert 0, so siehst du auf dem Bildschirm:

```
PRINTA-5
-5

READY.
```

Wir kommen hier in den Bereich der „negativen Zahlen“. Sie sind dadurch gekennzeichnet, daß vor ihnen stets ein Minuszeichen steht, z. B. [–] 5. Die Zahl –5 heißt dann auch „Minus Fünf“.

Das ist eigentlich ganz einfach zu erklären. Wenn man mehr wegnimmt, als da ist, also zum Beispiel von

0 etwas wegnehmen will, dann macht man sozusagen „Schulden“. Die negativen Zahlen sagen, wieviel man erst zurückgeben muß, damit man wieder auf 0 ist.

Oder ein anderes Beispiel: Auf einem Thermometer sind die Grad über Null und die Grad unter Null angezeichnet. Im Winter, wenn aus dem Wasser Eis und aus dem Regen Schnee wird, zeigt das Thermometer die Minus-Grade an. Das heißt: Soviel Grad muß es wärmer werden, soviel Grad müssen dazukommen, damit es wieder 0 Grad ist.

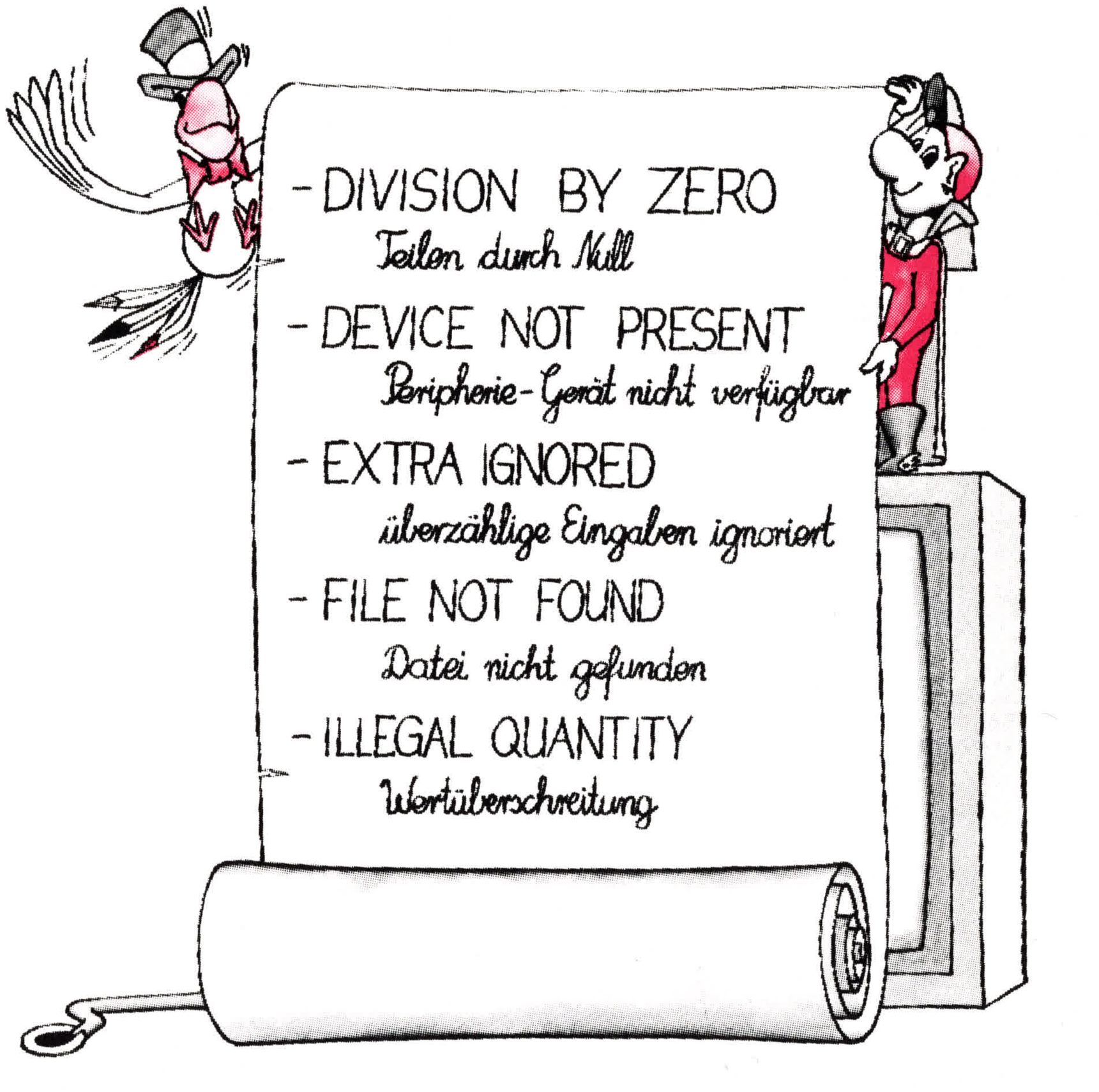
Noch etwas anderes kann passieren, falls unsere Variable den Wert 0 hat. (Fürs Ausprobieren kannst du ihn ja immer wieder eingeben – du weißt schon, mit

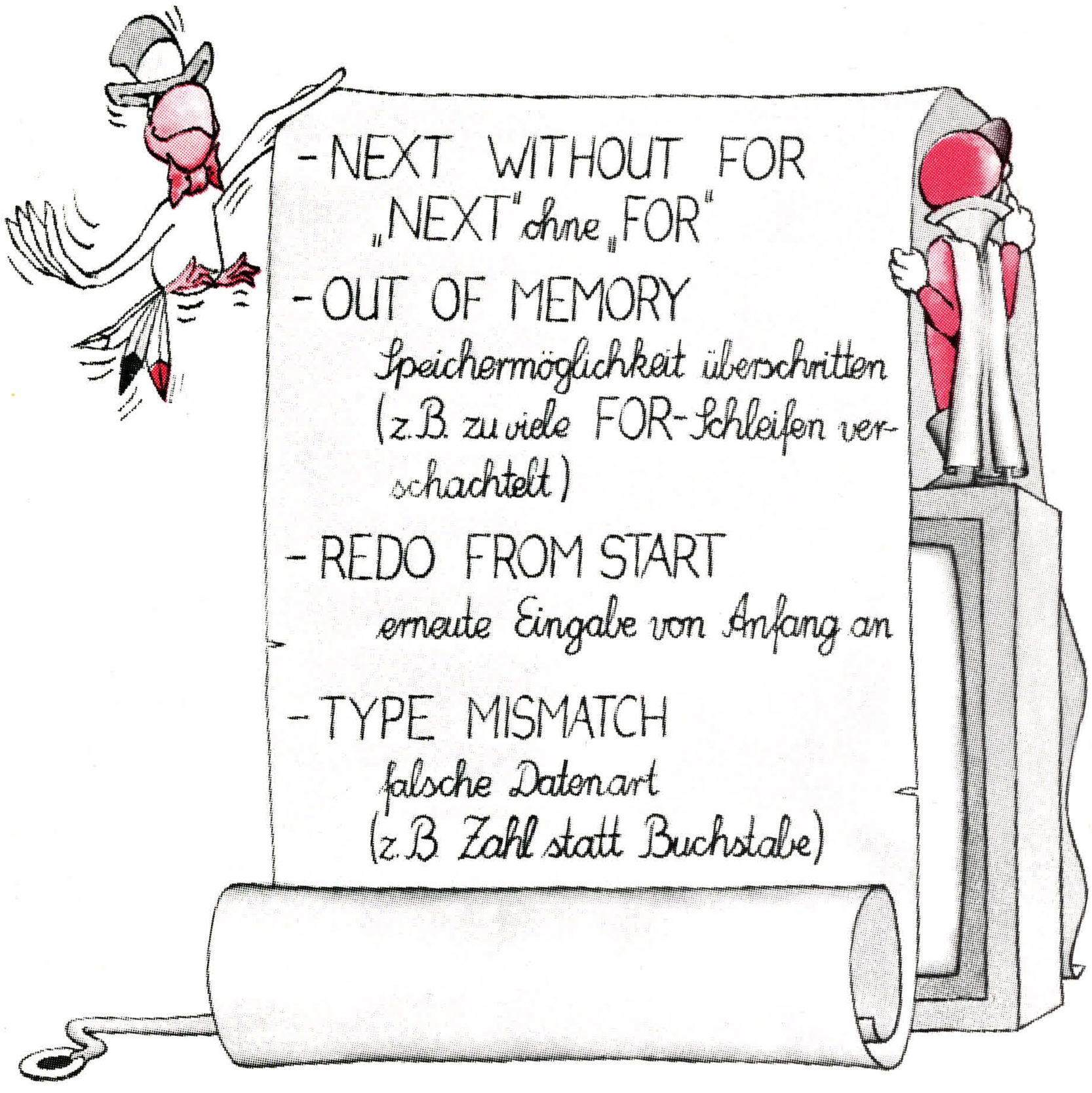
„A=0“.)

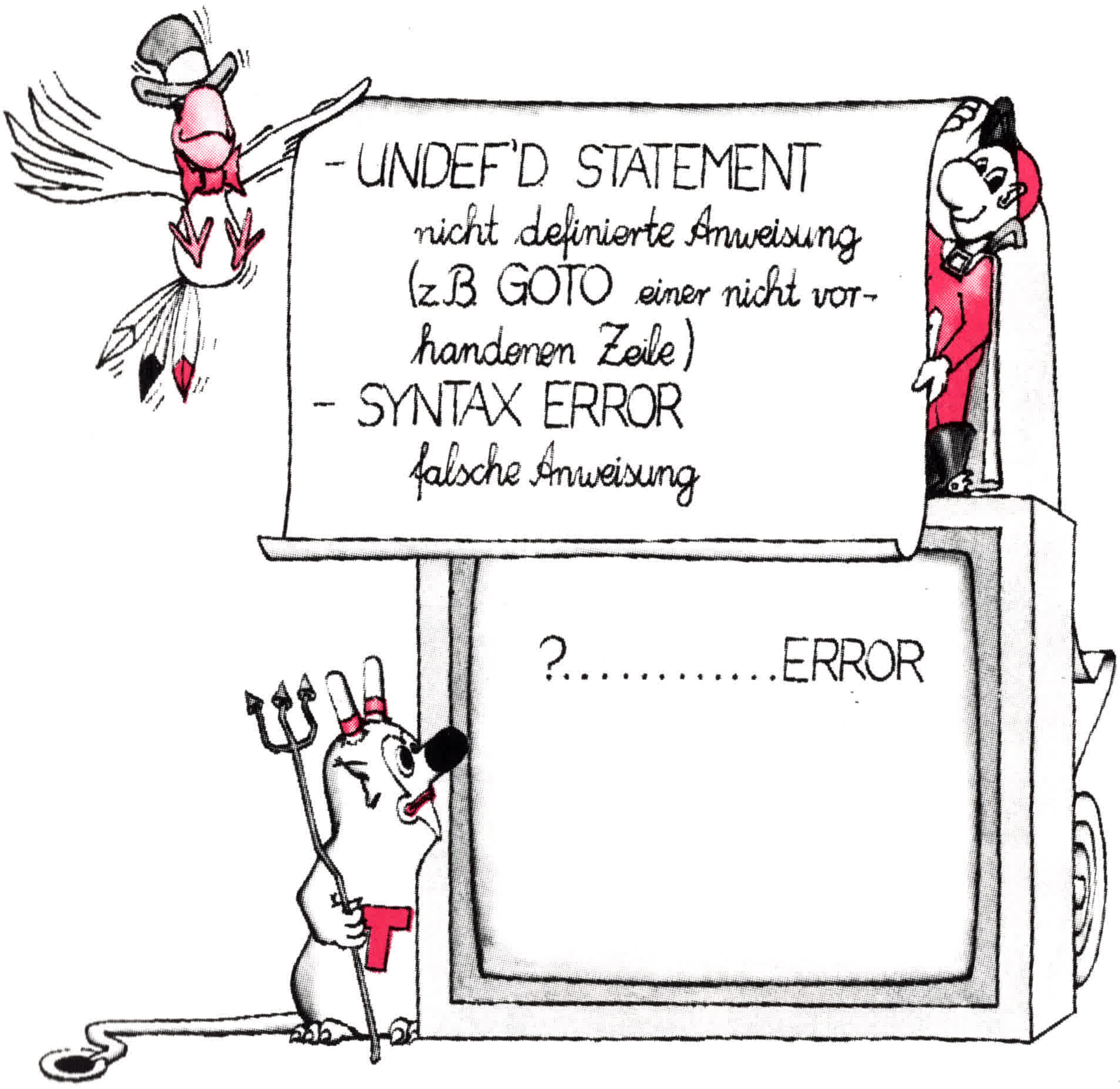
Nun stellen wir dem Computer die folgende Aufgabe:

```
PRINT2/A

?DIVISION BY ZERO  ERROR
READY.
```

Die Zeile mit dem Fragezeichen am Anfang ist eine Fehlermeldung des Computers. Jede Fehlermeldung beginnt mit einem Fragezeichen und endet mit dem Ausdruck ERROR, dem englischen Wort für Fehler. Eine Fehlermeldung macht der Computer, wenn er etwas, was man von ihm verlangt, nicht ausführen kann. In diesem Fall ist es die DIVISION BY ZERO, also das Teilen durch Null. Durch Null dividieren ist nämlich in der Mathematik unter gar keinen Umständen erlaubt. Und weil der Computer die Regeln der Mathematik in- und auswendig kennt, weigert er sich standhaft, irgendeine Zahl durch Null zu teilen. DIVISION BY ZERO wird übrigens so ausgesprochen: „diwischn bai siero“.


- DIVISION BY ZERO
Teilen durch Null
- DEVICE NOT PRESENT
Peripherie-Gerät nicht verfügbar
- EXTRA IGNORED
überzählige Eingaben ignoriert
- FILE NOT FOUND
Datei nicht gefunden
- ILLEGAL QUANTITY
Wertüberschreitung


- NEXT WITHOUT FOR
„NEXT" ohne „FOR"
- OUT OF MEMORY
Speichermöglichkeit überschritten
(z.B. zu viele FOR-Schleifen verschachtelt)
- REDO FROM START
erneute Eingabe von Anfang an
- TYPE MISMATCH
falsche Datenart
(z.B. Zahl statt Buchstabe)



Noch etwas kann beim Herumprobieren mit Zahlen und Variablen passieren: Bei Rechnungen, die nicht aufgehen, gibt der Computer die „Dezimal-Stellen“ hinter dem Komma an. Er setzt aber, wie du schon weißt, dafür einen Punkt. Das bedeutet nichts anderes als jene Hälften, Viertel, Drittel und so weiter, die beim Teilen übrigbleiben können und daher noch weiter aufgeteilt werden. Wenn beispielsweise vier Leute sich fünf Stück Kuchen gerecht teilen wollen, so müssen sie das letzte Stück noch einmal aufteilen. Vielleicht bleibt dabei wieder ein Stück übrig, und das muß dann noch mal geteilt werden. Und dann wird’s kompliziert. Aber unsere drei Freunde sind da natürlich nicht so pingelig.

Der Computer dagegen ist unerbittlich genau. Er gibt immer weiter die Teile an, und zwar in Zehnteln, in Hundertsteln, in Tausendsteln und so weiter. Das sind die Zahlen hinter dem Punkt.

Hier ist ein Beispiel:

```
A=6

READY.
B=7

READY.
PRINTA/B
 .857142857

READY.
```

Da wollten sich offenbar 7 Leute 6 Stück Kuchen teilen. Der Computer sagt ihnen, wie es gerecht wäre: Es bekommt nämlich keiner ein ganzes Stück. Nichts anderes sagt die leere Stelle vor dem Punkt. Beim Rechnen auf dem Papier schreibt man stattdessen oft eine Null, die der Computer nicht eigens anzeigt.

Jeder kriegt 8 Zehntel Teile von einem Stück Kuchen, 5 Hundertstel Teile, 7 Tausendstel Teile, 1 Zehntausendstel und so weiter. Das nächste sind dann schon so kleine Krümel, daß man sie mit dem Auge gar nicht mehr sehen würde. Aber der Computer nimmt es, wie gesagt, äußerst genau.

Das muß er allerdings auch. Denn mit der Hilfe eines Computers kann man auch wichtigere Dinge aufteilen als ein paar Stück Kuchen – zum Beispiel zusammengesetzte Stoffe in der Chemie. Auch beim Konstruieren von Häusern und Brücken ist die Genauigkeit des Computers mittlerweile unentbehrlich. Aber bestimmt wirst du selbst ebenfalls noch Programme kennenlernen, bei denen es auf sehr genaues Dividieren ankommt.

Die Rechenarten und -regeln, über die wir gesprochen haben, mußt du nicht gleich richtig „können“. Aber zumindest kannst du dir jetzt etwas darunter vorstellen, wenn so was mal vorkommt. Vor allem wirst du dann wohl kaum noch vermuten, daß dein Computer jetzt spinnt. Mitunter kann es dir so vorkommen, als wollte es dir der Computer besonders

schwer machen. Nun, abgesehen davon, daß eine Maschine keinen Willen hat: das „will“ er bestimmt nicht. Ein Computer kennt zwar genau die Rechenregeln, und er macht da meistens auch nichts falsch. Doch kann er nicht wissen, wie weit du, sein Programmierer, in Mathematik schon bist. Daher kann man sich auch beim Computernlernen nicht immer ganz genau daran halten, was in der Schule an Mathe schon durchgenommen worden ist und was nicht.

Mit unseren Variablen (so wollen wir ab jetzt unsere „Stellvertreter“ ein für allemal nennen) können wir noch viel mehr Dinge anfangen. Bis jetzt hätten wir statt der Variablen eigentlich ebensogut die entsprechende Zahl eintippen können. Die Programme wären dadurch keineswegs länger geworden.

Jetzt hingegen probieren wir mal, uns ein vollständiges Einmaleins ausgeben zu lassen – und zwar mit einem Programm, bei dem man nicht für jede Zahl eine neue Zeile benötigt. Der Trick dabei ist, daß wir ja mit unserem

GOTO

im Programm immer wieder an eine bestimmte Zeile zurückgehen können (natürlich auch vorwärts). Man nennt das eine Schleife. Bei den Schleifen mit dem GOTO-Befehl haben wir bis jetzt stets erreicht, daß der Computer etwas bei jedem Durchgang wieder

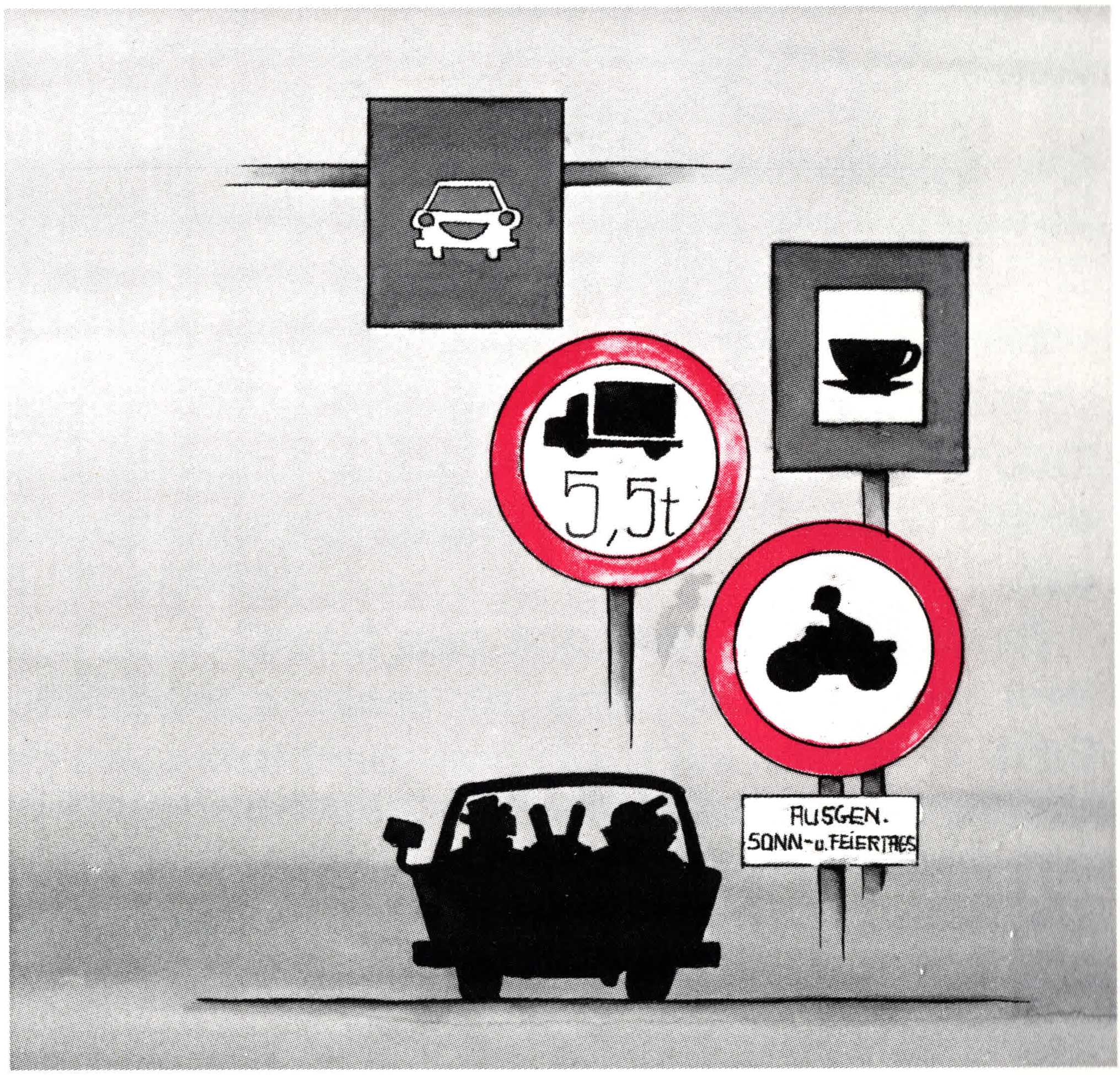
und immer wieder auf genau die gleiche Weise macht. Auch jetzt soll er etwas immer wieder tun, aber bei jedem „Durchlauf“ etwas anders. Du wirst gleich sehen, daß das gar nicht schwierig ist. Dieses

Programm sollst du nämlich selber schreiben. Ich erkläre nur Zeile für Zeile, was so in etwa passieren soll.

Zuvor mußt du nur noch zwei BASIC-Ausdrücke kennenlernen. Der eine ist etwas komplizierter. Dafür kann man auch sehr viel damit anfangen. Der andere ist ganz einfach.

Bis jetzt hat unser Computer immer sozusagen geradeaus gedacht. Wie ein Autofahrer, der auf einer geraden Straße fährt und sich daran hält, was

auf den Schildern steht. Ab und zu haben wir ihn auf einer Straße einfach im Kreis fahren lassen. Jetzt wollen wir ihn endlich einmal an eine Abzweigung kommen lassen. Dort soll er überlegen, in welche Richtung er weiterfahren muß. Dabei müssen wir ihm allerdings helfen. Wir stellen ihm eine Bedingung. Eine Bedingung ist sozusagen ein Grund, etwas zu tun. Ein bedingter Befehl ist also ein Befehl, der nur unter bestimmten Umständen gilt, nur dann nämlich, wenn eine Bedingung erfüllt ist. Wir schicken unseren Max als Autofahrer an eine Abzweigung.



Dort sieht er Schilder. Das sind bedingte Befehle. Und was er macht, das ist, logisch gesehen, eine „bedingte Abzweigung".

Die Bedingung des einen Schildes ist: *Wenn* du einen Lastwagen fährst, der mehr als fünfeinhalb Tonnen wiegt, mußt du einen anderen Weg nehmen.

Die Bedingung des anderen Schildes heißt: Wenn du ein Fahrzeug fährst, das für Autobahnen zugelassen ist, darfst du diesen Weg nehmen – sonst nicht.

Die Bedingung für das dritte Schild lautet: Mit einem Motorrad darfst du hier nicht durchfahren – außer an Sonntagen und Feiertagen.

Die Bedingung für das vierte Schild kannst du sicher selbst formulieren.

„*Wenn*" heißt im englischen „*if*". „*Dann*" heißt „*then*" („sen" mit dem stimmhaften Lispel-S gesprochen). Und aus eben diesen beiden englischen Wörtern besteht unser BASIC-Ausdruck:

IF . . . THEN

Zwischen IF und THEN steht, was die Bedingung sein soll. Der Ausdruck IF . . . THEN bedeutet demnach: *Wenn* (IF) etwas so und so ist (was, müssen wir dem Computer unbedingt sagen), *dann* (THEN) soll er etwas Bestimmtes tun (was, müssen wir ihm gleichfalls sagen). Am besten, wir schreiben wieder

ein kleines Programm.

Als erstes soll wieder ein „Sternchen-Film“ ablaufen. Das machen wir durch zwei Zeilen, etwa mit diesen:

```
10 PRINT,"■**"
20 PRINT,"■■**"
```

In der ersten Zeile bringt das Komma die Sternchen an den Beginn der zweiten Bildschirmspalte. Das Sonderzeichen vor den beiden Sternchen in der ersten Zeile entsteht durch die Cursor-Steuerung nach rechts, jedoch nur, falls die Cursor-Taste innerhalb der Anführungszeichen betätigt wird. In der Zeile 10 wird folglich das Sternchen um einen Zwischenraum nach rechts gerückt, in der zweiten um zwei Plätze nach rechts, jedoch erst, wenn es beim Programmdurchlauf auf dem Bildschirm ausgegeben wird. Jetzt fügen wir noch Zeile 30 hinzu. In ihr wollen wir eine Variable zählen lassen:

```
30 A = A + 1
```

Und in Zeile vierzig kommt unsere bedingte Verzweigung, damit der Computer weiß, wie oft er die vorherigen Programmschritte auszuführen hat: *Wenn* die Variable A den Wert 25 erreicht hat, *dann* soll der Computer zur Zeile 60 springen:

```
40 IF A=25 THEN GOTO 60
```

In Zeile 50 steht, was der Computer machen soll, solange diese Bedingung nicht erfüllt ist:

```
50 GOTO 10
```

Er soll zurück zur Zeile 10 springen. Dort soll er das Programm von vorn anfangen.

Jedesmal, wenn das Programm durch diese Schleife läuft, wird der Wert von A um 1 erhöht. *Wenn* A schließlich den Wert 25 erreicht hat, *dann* tritt die Bedingung in Kraft, und das Programm darf die Schleife verlassen. Es verabschiedet sich in Zeile 60:

```
60 PRINT "TSCHUESS !"
```

Mit Hilfe der Zeilen 30 und 40 haben wir uns einen „Rundenzähler" gebastelt: Zeile 30 zählt die Runden unserer Programmschleife, so ähnlich wie die Runden bei einem Rennen gezählt werden. Und in Zeile 40 steht, wann die Schleife, also das Rennen, zu Ende sein soll und was dann geschehen soll.

Das alles ist vielleicht zunächst ein wenig komplizierter als jene Dinge, die wir bis jetzt mit dem Computer angestellt haben. Jetzt ist es ja schon so, daß man beim Programmieren schon ein bißchen vorausdenken muß. Wir erleichtern uns die Arbeit, wenn wir uns vorher eine Skizze anfertigen.

Die Übersicht fällt uns leichter, wenn die einzelnen Namen für die Programmteile auch im Listing vor

kommen. Bei den etwas komplizierteren Programmen wollten wir doch sowieso REM-Zeilen als Überschriften verwenden. Der Ablauf ändert sich dadurch kein bißchen.

Der zweite BASIC-Befehl, den wir noch benötigen, ist sehr viel einfacher – wie versprochen. Er heißt

END

Was der bedeutet, ist ja wohl nicht allzu schwer herauszufinden. Eben – bei END hört der Computer nämlich mit der Programmdurchführung auf. Besonders interessant ist dieses BASIC-Wort zusammen mit der bedingten Verzweigung, also mit dem IF . . . THEN.

Nehmen wir ein besonders einfaches Beispiel und lassen den Computer bis 10 zählen.

Versuche doch, nach der folgenden Programmbeschreibung ein Programm zu schreiben:

10 Gib dem Programm einen Namen fürs Listing
20 Die Variable soll am Anfang den Wert eins haben, weil wir mit 1 zu zählen beginnen wollen.
30 Es soll der Wert ausgedruckt werden, den die Variable gerade hat
40 Die Variable soll nun um eins erhöht werden.
50 *Wenn* der Wert der Variablen 10 geworden ist,

dann soll das Programm zu Ende sein.
60 Sonst zurück in die Schleife: der nächste Wert für die Variable A soll ausgedruckt werden. Also geh zur Zeile – ja, zu welcher?

Dieses Programm selber zu schreiben, ist bestimmt nicht leicht. Aber bevor du auf der nächsten Seite nachsiehst, probiere es bitte aus. Laß deine Programme laufen, schau dir an, was auf den Bildschirm kommt, und überlege notfalls, wo vielleicht ein Fehler stecken könnte. Oder läuft's sowieso wunderbar? Um so besser!

```
10  REM########STERNCHENFILM########
20  PRINT,"■■"
30  PRINT,"■■■■"
40  REM RUNDENZAEHLER
50  A=A+1
60  IF  A=25  THEN  GOTO  100
70  REM SCHLEIFE
80  GOTO 10
90  REM ABSCHIED
100  PRINT "TSCHUESS!"

READY.
```

So *ungefähr* sollte also ein Programm zum Zählen aussehen. Wenn deines nicht haarscharf genauso aussieht, aber trotzdem richtig läuft – super!

Wahrscheinlich hattest du die meisten Probleme mit Zeile 50. Deine Zählmaschine hat bestimmt

zuerst bloß bis neun gezählt. Bei dieser Bedingung kommt das END nämlich genau dann, wenn sie erfüllt ist. Wir wollen das Ende aber erst haben, *nachdem* sie erfüllt ist.

Wenn du willst, kannst du nun noch ein paar ähnliche Programme zu schreiben versuchen. Dazu mußt du meistens nur eine oder zwei Zeilen verändern.

Hier einige Anregungen:

- Schreibe ein Programm, das die Zahlen von 1 bis 10 in eine Zeile bringt. (Hinweis: Denke an unsere verschiedenen Trennungszeichen!)
- Schreibe ein Programm, das von 10 bis 1 zurückzählt. (Hinweis: Zeile 20 und Zeile 40. Mehr sage ich nicht!)
- Schreibe ein Programm, das bei 2 anfängt und alle geraden Zahlen bis zwanzig zählt. (Hinweis: Drei Zeilen müssen geändert werden.)

Nachdem du diese drei Programme geschrieben hast, kannst du noch etwas Schwierigeres versuchen:

- Schreibe ein Programm, bei dem von 100 immer abwechselnd 3 abgezogen und 2 dazugezählt werden, bis 50 erreicht ist. Das Programm soll nicht einfach beendet werden, sondern sich mit einem Satz verabschieden.

Nach diesen Übungen können wir eigentlich mit dem Einmaleins-Programm anfangen. Falls es nicht sofort klappt – macht nichts. Mögliche Lösungen hat der Max immer anzubieten. Damit das Nachschauen aber nicht zu verführerisch ist, stehen sie auf dem Kopf und sind in Spiegelschrift geschrieben.

Dein Programm muß jedoch nicht haargenau wie das vom Max aussehen. In BASIC gibt es ja, wie wir gesehen haben, oft verschiedene Möglichkeiten, etwas zu erreichen.

```
10  REM +++ ZAEHLMASCHINE +++
20  A=10
30  PRINT A;
40  A=A-1
50  IF  A=0   THEN END
60  GOTO 30
```

```
10  REM +++ ZAEHLMASCHINE +++
20  A=1
30  PRINT A;
40  A=A+1
50  IF  A=11  THEN END
60  GOTO 30
```

```
10  REM +++ ZAEHLMASCHINE +++
20  A=2
30  PRINT A;
40  A=A+2
50  IF  A=0   THEN END
60  GOTO 30
```

```
10 REM +++ ZAEHLMASCHINE +++
20 A=100
30 PRINT A;
40 A=A+2
50 PRINT A;
60 A=A-3
70 IF A=49 THEN GOTO 90
80 GOTO 30
90 PRINT:PRINT,"SO EIN ZAHLENSALAT!"

RUN
 100   102   99   101   98   100   97   99   96
 98    95    97   94    96   93    95   92   94   91
 93    90    92   89    91   88    90   87   89   86
 88    85    87   84    86   83    85   82   84   81
 83    80    82   79    81   78    80   77   79   76
 78    75    77   74    76   73    75   72   74   71
 73    70    72   69    71   68    70   67   69   66
 68    65    67   64    66   63    65   62   64   61
 63    60    62   59    61   58    60   57   59   56
 58    55    57   54    56   53    55   52   54   51
 53    50    52
          SO EIN ZAHLENSALAT!
READY.
```

Und so könnte unser „Standard"-Einmaleins-Rechner aussehen:

Wir beginnen wieder mit einer Zeile, die den Namen des Programms im Listing vermerkt, ohne den Programmablauf zu beeinflussen. Und vielleicht noch eine zweite Zeile, in der steht, wer es geschrieben hat. Du natürlich.

Jetzt wird's schon schwieriger. In der einen Zeile müssen wir festlegen, mit welcher Zahl das Einmaleins ausgeführt werden soll. Soll es das Dreier-Einmaleins sein, so bekommt unsere erste Variable den Wert drei; soll es das Vierer-Einmaleins sein, so bekommt sie den Wert vier, soll es das Zwölfer-Einmaleins sein, so bekommt sie den Wert zwölf; und so weiter . . .

Wir könnten hier statt dessen einfach eine Zahl einsetzen. Aber wenn in dieser Zeile eine Variable definiert wird (so nennt man es, wenn man ihr einen Zahlen-Wert zuordnet), dann genügt es später, wenn nur diese Zeile geändert wird. Und schon haben wir dann ein neues Einmaleins-Programm.

Nun wird die zweite Variable definiert. Das ist die Zahl, mit der unsere Einmaleins-Zahl später malgenommen, also multipliziert werden soll. Später muß sie in unserer Schleife ihren Wert erhöhen. Anfangen wollen wir, wie es sich für ein Einmaleins gehört, mit der 1. Diese Variable darf nicht mit demselben Buchstaben benannt werden wie die erste.

In der nächsten Zeile folgt die eigentliche Aufgabe für den Computer. Er soll multiplizieren, und zwar die

Zahlen, die wir jeweils durch unsere Variablen ausdrücken. Mit Hilfe des Trennungszeichens hinter dieser Aufgabe kann man die Ergebnisse hintereinander in einer Zeile oder in Spalten ausdrucken lassen – oder in einer einzigen Spalte, falls man gar kein Zeichen verwendet.

In der nächsten Zeile soll die Zahl, mit der unsere Einmaleins-Zahl multipliziert wird, um 1 erhöht werden. Klar, das Einmaleins geht ja auch dementsprechend: 1 mal A (unsere Einmaleins-Zahl), 2 mal A, 3 mal A, und so weiter. Würden wir nun schon in der nächsten Zeile unsere Schleife einbauen, dann würde das Programm laufen und laufen und laufen, bis zu einem Punkt, wo sogar unser Computer nicht mehr weiterrechnen kann. Deshalb müssen wir in der nächsten Zeile festlegen, wann der Computer mit dem Ausrechnen des Einmaleins aufhören soll.

Einigen wir uns auf 20? In dieser Zeile sollte also stehen: Wenn die Zahl, mit der unsere Einmaleins-Zahl multipliziert werden soll, größer als 20 wird, dann soll das Programm zu Ende sein. In der nächsten Zeile können wir nunmehr unsere Schleife aufbauen. Wir schicken den Computer zu der Zeile zurück, in der er rechnet.

Damit sollte das Programm einwandfrei laufen können. Wenn du dein Programm zum Laufen gebracht hast, dann hast du etwas hinter dir, was jeder Programmierer tut, egal wie kompliziert seine Programme aufgebaut sind: Du hast eine Aufgabe (nämlich ein Einmaleins) für den Computer in einzelne Teile „zerlegt“ und diese Teile dann in die Sprache des Computers, nämlich in BASIC, übertragen. Und so etwa sollte das Programm aussehen:

```
10 REM**EINMALEINS**
20 REMVON
30 A=4
40 B=1
50 PRINTA*B,
60 B=B+1
70 IFB=21THEM END
80 GOTO50
```

Aus diesem Programm läßt sich mit einigen zusätzlichen Zeilen ein Einmaleins-Übungs-Programm entwickeln, bei dem der Computer dich „abfragen“ kann. Doch jetzt kannst du zuerst einmal unseren Freund Max auf Luftballon-Jagd schicken! (Siehe Seite 94).

Tips & Tricks für das Eingeben der Listings

Beim Abtippen der Listings schleichen sich gern Fehler ein. Die folgenden Tips sollen dich davor bewahren und helfen, Fehler schnell zu finden.

TIP 1
Lege das Listing, das du eintippen willst, so vor dich hin, daß du es gut lesen kannst. Als Lesehilfe kannst du ein unbeschriebenes Blatt Papier verwenden. Lege es genau unter die Zeile, die du als nächstes eingeben willst.

TIP 2
Hake mit einem Stift jede eingegebene Zeile ab. So gibst du keine Zeile doppelt ein.

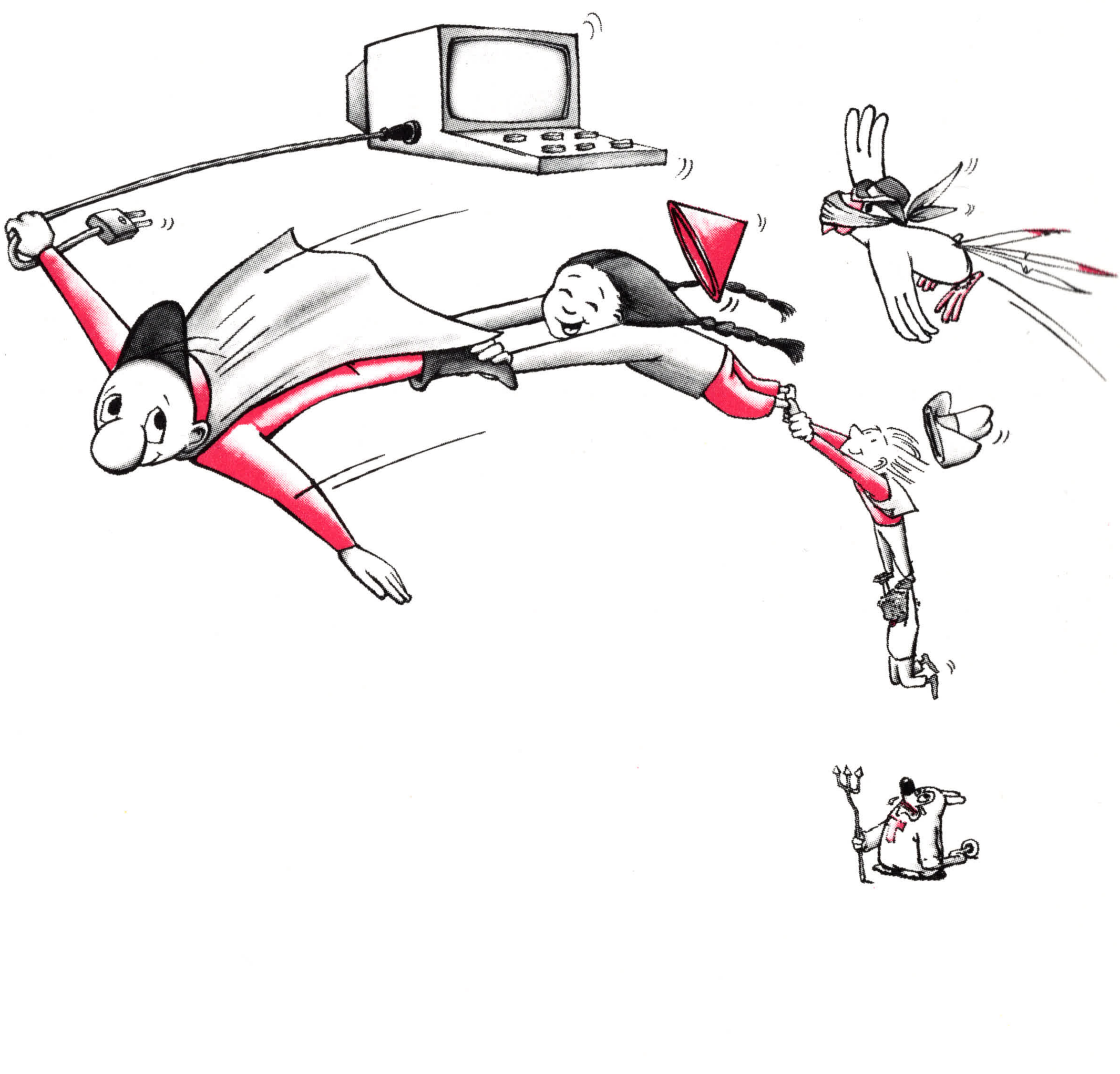
TIP 3
Speichere dein Programm des öfteren mal ab, zum Beispiel alle 10 Zeilen. Sollte inzwischen der Rechner abstürzen, gehen nur die Zeilen verloren, die du noch nicht abgespeichert hast.

TIP 4
Spätestens wenn du das eingegebene Programm starten willst, solltest du das Programm zuvor abspeichern. Denn eventuelle Fehler können den Rechner so blockieren, daß du nur durch Ausschal-

ten des Rechners wieder in den normalen Betrieb gelangst, und das Programm wäre damit verloren. Sobald du das Programm abgespeichert hast, kannst du es wieder in den Rechner laden und den Fehler suchen.

TIP 5

Falls du mit BASIC schon einigermaßen vertraut bist, noch ein weiterer Tip: Wenn Verzweigungen kommen (GOTO, GOSUB), mach's so, wie der Rechner später, „springe" in diese Verzweigungen und tippe diese zuerst ab, bevor du mit dem Hauptprogramm weitermachst. Das hat den Vorteil, daß du jetzt schon Probestarts des Programmes machen, und schon etwaige Fehler ausbessern kannst. Beachte dabei TIP 4.



Was heißt was in BASIC

Die wichtigsten BASIC-Begriffe aus diesem Buch

So heißt es in BASIC:	**Und das soll der Computer tun:**
AND	Lasse bei IF . . . THEN mehrere Bedingungen gelten.

Beispiel

```
10 INPUT A
20 INPUT B
30 PRINT A+B
40 A=A+1
50 B=B+2
60 IF A>10 AND B>20THEN END
70 GOTO 30
```

END	Beende das Programm
GOTO	Gehe zur Zeile deren Nummer nach GOTO kommt

Beispiel:

```
10 PRINT "HALLO"
20 PRINT "COMPUTER"
30 GOTO 10
```

IF . . . THEN	WENN eine Bedingung erfüllt ist, DANN springe zur Zeile, die nach THEN steht

Beispiel:

```
10 INPUT A
20 IF A>10 THEN GOTO 10
30 IF A<10 THEN PRINT "NA ENDLICH"
40 END
```

INPUT

Frage erst nach einer Eingabe (zum Beispiel einer Zahl oder einem Buchstaben), bevor du ein Programm weiter bearbeitest.

Beispiel:

```
10 INPUT A
20 PRINT "A*10=";A*10
30 GOTO 10
40 END
```

LIST

Zeige das Programm-Listing auf dem Bildschirm.

LOAD

Lade ein Programm vom Datenträger (Kassette oder Diskette)

Beispiel:

LOAD „Leiterspiel“ lädt ein Programm mit dem Namen „Leiterspiel“ von der Kassette. Auf der Datasette oder dem Kassettenrecorder wird PLAY gedrückt.

LOAD „Leiterspiel“, 8 lädt das Programm „Leiterspiel“ von der Diskettenstation.

NEW — Lösche dein Gedächtnis.

OR — Lasse bei IF . . . THEN mehrere Bedingungen gelten.

Beispiel:

```
10 INPUT A
20 INPUT B
30 PRINT A+B
40 A=A+1
50 B=B+2
60 IF A=10 OR B>20THEN END
70 GOTO 30
```

PRINT — mit Anführungszeichen: Schreibe oder zeichne auf den Bildschirm, was zwischen den Anführungszeichen steht.

Beispiel:

```
10 PRINT"MAX"
```

PRINT — ohne Anführungszeichen: Führe aus, rechne aus, was hinter dem PRINT steht.

Beispiel:

```
10 PRINT 23-9
```

REM — Schreibe eine Bemerkung des Programmierers in das

Listing. Sie soll das Programm selbst nicht beeinflussen.

Beispiel:

```
10 REM *ZWEIER-EINMALEINS*
20 INPUT A
30 PRINT A*2
40 GOTO 20
```

RETURN

Funktionstaste:
Merke dir, was ich gerade geschrieben habe und gehe mit dem Cursor an den Anfang der nächsten Zeile.

RUN

Führe das Programm aus.

RUN/STOP

Funktionstaste:
Halte das Programm dort an, wo du gerade bist und melde mir mit BREAK IN . . . (Zeilennummer) wo die Unterbrechung stattgefunden hat.

SAVE

Speichere ein Programm auf dem Datenträger (Kassette oder Diskette) ab.

Beispiel:

SAVE "Leiterspiel" speichert ein Programm auf Kassette. Es müssen die beiden Tasten PLAY und RECORD gedrückt werden.

Save "Leiterspiel", 8 speichert das Programm „Leiterspiel" auf Diskette.

Und die Steuerzeichen:

"▶" Cursor um eine Stelle nach rechts.

"◀" Cursor um eine Stelle nach links.

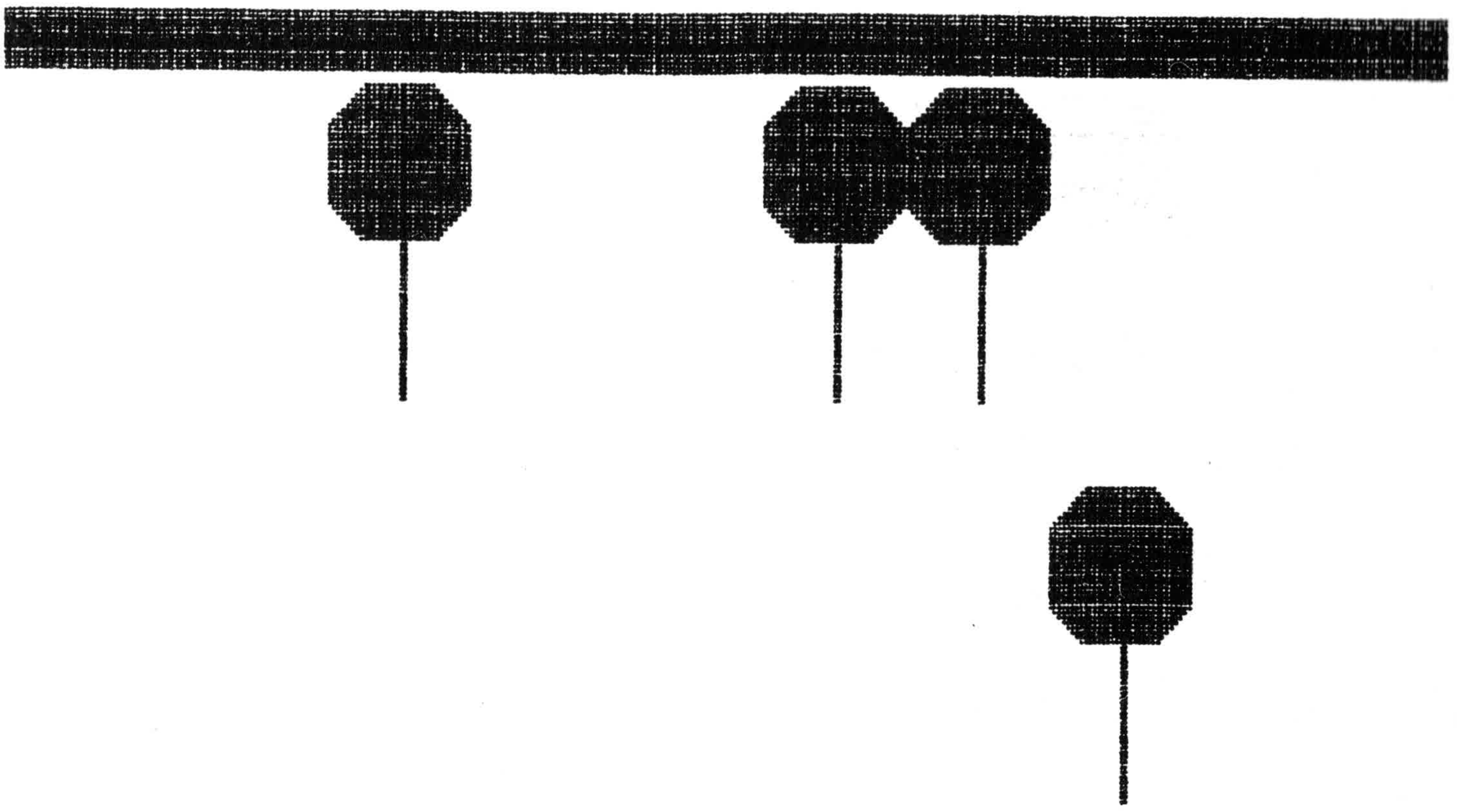
"▲" Cursor um eine Zeile nach oben.

"▼" Cursor um eine Zeile nach unten.

"■" Bildschirm löschen.

BALLONJAGD

In diesem Spiel soll der Computer-Max zuerst ein paar Luftballons zum Platzen bringen. Natürlich mußt du ihm dabei helfen. Hinter den Luftballons kommen Zahlen zum Vorschein, die zusammen eine Rechenaufgabe ergeben.

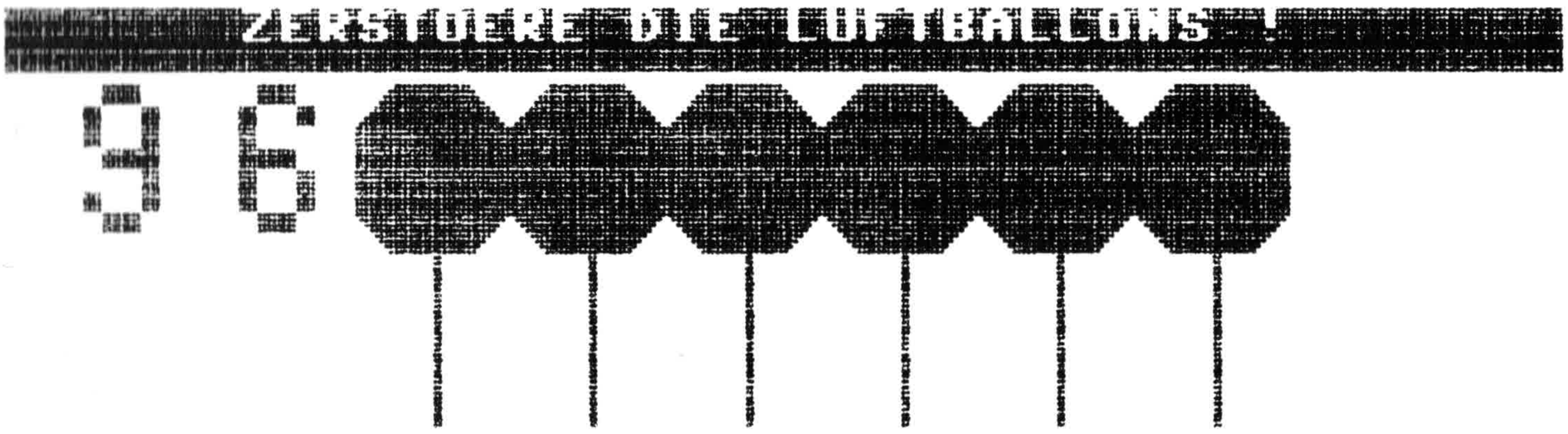


Nachdem die bunten Luftballons aufgestiegen sind, erscheint die Aufforderung: „Zerstöre die Luftballons!“ Jetzt mußt du den Max steuern.

Mit [?/] macht er einen Schritt nach rechts

Mit [Z] macht er einen Schritt nach links

Mit der **Leertaste** springt er nach oben und läßt einen Luftballon platzen.



Wenn alle Ballons geplatzt sind und die Rechnung vollständig zu sehen ist, kehrt der Max um und geht an den Anfang einer Zahlenreihe von 0 bis 9. Du steuerst ihn über eine Zahl und drückst die Leertaste, um sie in die Zeile zu bringen, in der steht: „Das Ergebnis lautet:“

Aber Achtung! Die Zahlen werden wie beim schriftlichen Zusammenzählen von hinten eingegeben. Man fängt also mit der letzten Zahl an.

Max hat auch einen **Radiergummi** zur Verfügung, mit der er jeweils die Zahl wieder löschen kann, die er zuletzt eingegeben hat. Dazu muß er über dem Radiergummi stehen.

Wenn du denkst, dein Ergebnis ist richtig, dann drücke RETURN.

Der Computer sagt dir, ob das Ergebnis richtig oder falsch ist. Wenn es richtig ist, freut sich der Max. Der Computer fragt, ob du noch ein Spiel machen willst.

Sobald das Ergebnis falsch ist, sagt der Computer, ob es zu groß oder zu klein geraten ist. Dann kannst du es mit Hilfe von Max ausradieren, um es noch einmal zu versuchen.

```
100 REM ------------------
110 REM ------------------
120 REM ## BALLONSPIEL ##
130 REM ------------------
140 REM ------------------
150 DIMZ$(10):POKE788,49:POKE792,71
160 L$="                                                                                ":ER=10
170 Q$="                ":Y=150:POKE53269,0
180 ZE$="                                        "
190 L1$="                                                                                "
200 ZA$(0)="  "+Q$+"  "+Q$+"  "+Q$+"    "
210 ZA$(1)="  "+Q$+"  "+Q$+"  "+Q$+"  "
220 ZA$(2)="  "+Q$+"  "+Q$+"  "+Q$+"  "
230 ZA$(3)="  "+Q$+"  "+Q$+"  "+Q$+"    "
240 ZA$(4)="  "+Q$+"  "+Q$+"  "+Q$+"  "
250 ZA$(5)="  "+Q$+"  "+Q$+"  "+Q$+"    "
260 ZA$(6)="  "+Q$+"  "+Q$+"  "+Q$+"    "
270 ZA$(7)="  "+Q$+"  "+Q$+"  "+Q$+"  "
280 ZA$(8)="  "+Q$+"  "+Q$+"  "+Q$+"    "
290 ZA$(9)="  "+Q$+"  "+Q$+"  "+Q$+"    "
300 LE$="    "+Q$+"    "+Q$+"    "+Q$+"      "
310 PL$="    "+Q$+"  "+Q$+"  "+Q$+"      "
320 R$(8)="    "+Q$+"  "+Q$+"  "+Q$+"      "
330 ZA$(10)="  "+Q$+"  "+Q$+"  "+Q$+"    "
340 A$(1)="    "
350 A$(2)="    "
360 A$(3)="    "
370 A$(4)="    "
380 A$(5)="    "
```

```
390 A$(6)="  |  "
400 A$(7)="  |  "
410 A$(8)="  |  "
420 A$(9)="     "
430 A(0)=28:A(1)=30:A(2)=31:A(3)=158:A(4)=150:A(5)=159:A(6)=156:A(7)=154:A(8)=12
9
500 REM ------------------------------
510 REM ## LUFTBALLONS STEIGEN AUF ##
520 REM ------------------------------
530 POKE53280,6:POKE53281,1:PRINT"▚▄▟                                                    ";
540 PRINT"____________________________________________________▄"
550 A=INT(RND(1)*8)*4+5
560 I=0
570 IFB(I)=ATHEN550
580 IFI<=BTHENI=I+1:GOTO570
590 B(B)=A:B=B+1
600 C=INT(RND(1)*9)
610 FORI=23TO1STEP-1
620 A$=CHR$(A(C))
630 JM=24-I:IFJM>9THENJM=9
640 FORJ=1TOJM:A$=A$+"▚▄▄▄▄"+A$(J):NEXT
650 PRINT"▟"TAB(A)LEFT$(ZE$,I)A$;
660 NEXT
670 C$((A-5)/4)=A$
680 IFB<8THEN550
1000 REM ----------------
1010 REM ## RECHNUNGEN ##
1020 REM ----------------
1030 ONINT(RND(0)*1+1)GOSUB1130,63999,63999,63999
```

```
1040 FORI=1TO7
1050 I$=MID$(C$,I,1)
1060 IFI$=)"0"ANDI$<="9"THENR$(I)=ZA$(VAL(I$))
1070 IFI$="+"THENR$(I)=PL$
1080 IFI$=" "THENR$(I)=LE$
1090 NEXT:LB=0:PRINT"■■      ZERSTOERE DIE LUFTBALLONS !""
1100 GOTO 2000
1110 FORI=1TO7:PRINT"■"TAB(I*4-3)R$(I):NEXT:PRINT"■■■■■■■■"
1120 END
1130 A1=INT(RND(0)*998+1):A2=INT(RND(0)*998+1):C=A1+A2
1140 A$=RIGHT$("   "+STR$(A1),3):B$=RIGHT$("   "+STR$(A2),3)
1150 C$=A$+"+"+B$
1160 RETURN
2000 REM ------------------------------
2010 REM ## ZERSTOEREN DER BALLONS ##
2020 REM ------------------------------
2030 FORI=0TO190:READX:POKE832+I,X:NEXT
2040 V=53248:X=16:X1=16:POKEV,X:POKEV+1,Y:POKEV+21,1:POKEV+16,0
2050 POKEV+23,7:POKEV+29,7
2060 POKE2040,13:POKE2041,14:POKE2042,15
2070 POKEV+39,9:POKEV+40,9:POKEV+41,9:REM FARBE
2080 GETA$:IFQY>14THEN3010
2090 IFA$=""THEN2080
2100 IFA$="Z"THENX=X-24:GOSUB5040
2110 IFA$="/"THENX=X+24:GOSUB5040
2120 IFA$=" "THENGOSUB5170
2130 GOT02080
3000 REM ----------------------
3010 REM ## ERGEBNISEINGABE ##
```

```
3020 REM ---------------------
3030 XM=X1:FORI=1TO9:GETA$:NEXT
3040 FORX=XMTO24STEP-16
3050 GOSUB5040:NEXTX
3060 PRINT"S"TAB(34)"■QQR▀▄QR▀▄QL"
3070 PRINT"SQ GEBE JETZT DAS RICHTIGE ERGEBNIS EIN !▀":SU=0
3080 Y=103:POKEV,40:POKEV+1,Y:X=40:X1=X
3090 FORI=0TO10:PRINT"S"TAB(I*3+3)"QQQQQQQQQQQQ■"ZA$(I):NEXT
3100 PRINT"■"L1$L1$L1$L1$
3110 PRINT"QQ DAS ERGEBNIS LAUTET:"
3120 GETA$:IFA$=""THEN3120
3130 IFA$=CHR$(13)ORER=5THEN4010
3140 IFA$="Z"THENX=X-24:X2=-1:GOSUB5040
3150 IFA$="/"THENX=X+24:X2=1:GOSUB5040
3160 IFA$=" "THENX2=0:GOSUB5170:GOSUB3220:GOTO3090
3170 GOTO3120
3180 POKEV+31,0:QY=QY+1
3190 RX=INT((X-16)/32)+1
3200 PRINT"S"TAB(RX*4-3)"QQ■"R$(RX)"Q■I QI QI QI "
3210 RETURN
3220 RX=INT((X-18)/24):IFRX=10ANDER<10THENER=ER+1:S(10-ER)=0
3230 FORI=12TO15
3240 PRINT"S"TAB(RX*3+3)LEFT$(ZE$,I)"    Q■■I"ZA$(RX):NEXT
3250 IM=1:IFRX*3+3>ER*3+3THENIM=-1
3260 FORI=RX*3+3TOER*3+3STEPIM
3270 PRINT"SQQQQQQQQQQQQQQQQQQ"RIGHT$(L$,I+4)L$L$L$LEFT$(L$,36-I);
3280 PRINT"]]]]"TAB(I)ZA$(RX)
3290 NEXT
3300 FORI=16TO20
```

```
3310 PRINT"■"TAB(ER*3+3)LEFT$(ZE$,I)"   ■■■■"ZA$(RX);
3320 NEXT
3330 IFRX=10THENPRINT"■"TAB(ER*3+3)LEFT$(ZE$,21)"   ■■■■   ■■■■   ■■■■   ";
3340 IFRX<10THENER=ER-1:S(9-ER)=RX
3350 RETURN
4000 REM ------------------------
4010 REM ## ERGEBNISKONTROLLE ##
4020 REM ------------------------
4030 SU=S(4)*10000+S(3)*1000+S(2)*100+S(1)*10+S(0)
4040 IFSU<>CTHEN4180
4050 PRINT"■■■     DEIN ERGEBNIS IST RICHTIG!"
4060 XM=X1
4070 FORX=XMTO40STEP-16:GOSUB5040:NEXT
4080 FORY=103TO193STEP10:GOSUB5040:NEXT
4090 FORXM=0TO5
4100 PRINT"■"TAB(16)"■■■■■■■■■■■■■■JIPPEE !":GOSUB5170
4110 PRINT"■"TAB(16)"■■■■■■■■■■■■■■          ":FORJ=0TO200:NEXT
4120 NEXT
4130 PRINT"■■    WILLST DU NOCH EINMAL SPIELEN ?"
4140 PRINT"■  GEBE J FUER JA ODER N FUER NEIN EIN !"
4150 GETA$:IFA$="J"THENRUN
4160 IFA$<>"N"THEN4150
4170 PRINT"■":FORI=1TO16:POKEV+I,0:NEXT:END
4180 IFSU>CTHENPRINT"■■      DEIN ERGEBNIS IST ZU GROSS !      "
4190 IFSU<CTHENPRINT"■■      DEIN ERGEBNIS IST ZU KLEIN !      "
4200 GOSUB4260:XM=X1
4210 FORX=XMTO280STEP24:GOSUB5040:NEXT:GOSUB4260
4220 X=280:X1=X:IFER=10THEN3010
4230 GOSUB5170:GOSUB3220:GOSUB4260
```

```
4240 FORI=0TO10:PRINT"S"TAB(I*3+3)"QQQQQQQQQQQQ■"ZA$(I):NEXT
4250 PRINTL1$L1$L1$L1$:GOTO4220
4260 POKE55296,1
4270 FORI=1TO39:POKE55296+I,1:POKE55295+I,0:NEXT
4280 POKE55296+39,0:RETURN
5000 REM --------------------
5010 REM ## UNTERPROGRAMME ##
5020 REM --------------------
5030 REM ----------
5040 REM # LAUFEN #
5050 REM ----------
5060 IFX<16THENX=16:RETURN
5070 IFX>280THENX=280:RETURN
5080 X2=(X+X1)/2
5090 FORI=0TO20:NEXT
5100 POKEV+16,7*INT(X2/256)
5110 POKEV+2,X2-INT(X2/256)*256:POKEV+3,Y-2:POKEV+21,2
5120 FORI=0TO70:NEXT
5130 POKEV+16,7*INT(X/256)
5140 POKEV,X-INT(X/256)*256:POKEV+1,Y:POKEV+21,1
5150 X1=X:RETURN
5160 REM ------------
5170 REM # SPRINGEN #
5180 REM ------------
5190 FORI=1TO23
5200 Y1=50*SIN(π*I/24)
5210 POKEV+4,X-INT(X/256)*256:POKEV+5,Y-Y1:POKEV+21,4
5220 IFPEEK(V+31)>0THENGOSUB3180
5230 FORJ=0TO10:NEXT
```

```
5240 NEXTI
5250 POKEV+21,1
5260 RETURN
6000 REM ------------------
6010 REM # STEHENDER MAX #
6020 REM ------------------
6030 DATA 000,056,000,000,124,000,000,127,000,000,084,000,000,124,000
6040 DATA 000,056,000,000,016,000,000,124,000,000,238,000,000,254,000
6050 DATA 000,238,000,000,254,000,000,186,000,000,254,000,000,124,000
6060 DATA 000,108,000,000,108,000,000,108,000,000,108,000,001,239,000
6070 DATA 001,239,000,000
6080 REM --------------------
6090 REM # SPRINGENDER MAX #
6100 REM --------------------
6110 DATA 000,056,000,000,124,000,000,127,000,000,084,000,000,124,000
6120 DATA 000,056,000,000,016,000,000,254,000,001,239,000,003,125,128
6130 DATA 006,108,192,012,124,096,008,056,032,000,124,000,000,124,000
6140 DATA 000,198,000,001,131,000,003,001,128,006,000,192,030,000,240
6150 DATA 030,000,240,000
6160 REM --------------------
6170 REM # SPRINGENDER MAX #
6180 REM --------------------
6190 DATA 000,056,000,000,124,000,024,127,048,024,084,048,012,124,096
6200 DATA 006,056,192,003,017,128,001,255,000,000,238,000,000,124,000
6210 DATA 000,108,000,000,124,000,000,056,000,000,124,000,000,124,000
6220 DATA 000,198,000,001,131,000,003,001,128,006,000,192,030,000,240
6230 DATA 030,000,240

READY.
```

```
1 REM "┌┬┬──────────────────┬┬┐
2 REM "││└────────────────┘││
3 REM "│└──────────────────┘│
4 REM "│RECHENMAMPFER│
5 REM "│┌──────────────────┐│
6 REM "││┌────────────────┐││
7 REM "└┴┴──────────────────┴┴┘
10 DIMA$(24,2)
20 S1$="▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌"
30 S2$="▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯"
100 REM -----------------------
110 REM ### ECHTE ZUFALLSZAHL ###
120 REM -----------------------
130 POKE139,PEEK(56324):POKE140,PEEK(56325)
140 POKE141,PEEK(160):POKE142,PEEK(161):POKE143,PEEK(162)
200 REM ---------------
210 REM ### SPIELFELD ###
220 REM ---------------
230 PRINT"▚▖";:POKE53280,6:POKE53281,1
240 FORI=1TO24:READA$(I,1):NEXT
250 FORI=1TO24:READA$(I,2):NEXT
260 FORI=1TO24:PRINTA$(I,2)A$(I,1);:NEXT
270 GOSUB4110
300 REM ---------------------
310 REM ### SPRITE EINLESEN ###
320 REM ---------------------
330 V=53248:POKEV+0,0
340 POKEV+21,1:REM AKTIVIEREN
350 POKE2040,13
```

```
360 FORI=0TO62:READX:POKE832+I,X:NEXT
370 POKEV+39,5:POKEV+16,0:POKEV+23,0:POKEV+29,0
400 REM ---------------------
410 REM ## RECHEN VORGANG ##
420 REM ---------------------
430 FORI1=1TO10:REM AUFGABENANZAHL
440 A$="NR.:"+STR$(I1):S1=27:S2=8:GOSUB4200
450 A$="RECHNE":S1=27:S2=9:GOSUB4200
460 A$="BITTE AUS:":S1=27:S2=10:GOSUB4200
470 A=INT(RND(0)*9+1.5)
480 B=INT(RND(0)*9+1.5)
490 C=INT(RND(0)*9+1.5)
500 ONINT(RND(0)*3.9+1)GOSUB1000,1110,1210,1310,63999,63999
510 S1=27:S2=12:GOSUB4200
520 GOSUB4310
530 A$=RIGHT$("    "+RE$,4):S1=29:S2=14:GOSUB4200
540 RA=VAL(RE$)
550 IFRA<>RETHEM710
560 A$="RICHTIG !":S1=28:S2=16:GOSUB4200
570 A$="NUN DARFST DU":S1=26:S2=18:GOSUB4200
580 A$="30SEK SPIELEN":S1=26:S2=19:GOSUB4200
590 IFA1=1THENGOSUB2070:GOTO900
600 IFA1=0THENA1=1:GOSUB2010
610 GOTO900
700 REM --------------------
710 REM # FALSCHE ANTWORT #
720 REM --------------------
730 A$="FALSCH !":S1=28:S2=16:GOSUB4200
740 IFRA<RETHENA$=" ZU NIEDRIG "
```

```
750 IFRA>RETHENA$="  ZU HOCH   "
760 S1=26:S2=18:GOSUB4200
770 A=ABS(RE-RA):P1=P1-A:GOSUB4110
780 A$="MINUSPUNKTE:":S1=26:S2=21:GOSUB4200
790 A$="    "+STR$(A):S1=26:S2=22:GOSUB4200
800 FORI=0TO4000:NEXT
810 A$="              ":S1=26:S2=21:GOSUB4200
820 A$="              ":S1=26:S2=22:GOSUB4200
830 A$="VERSUCHS":S1=28:S2=16:GOSUB4200
840 A$=" NOCH MAL ":S1=27:S2=18:GOSUB4200
850 GOTO520
900 FORI=1TO4000:NEXT
910 PRINT"■";
920 FORI=1TO24:PRINTLEFT$(S1$,25)A$(I,1);:NEXT
930 NEXT
940 POKEV,0:PRINT"■"
950 PRINT"■■■■■■■■■■■■■    DEINE GESAMTPUNKTZAHL BETRAEGT:"
960 PRINT"■              "P+P1"PUNKTE":I=0
970 PRINT"■   NOCHMAL SPIELEN? (J/N)"
980 GETA$:IFA$<>"J"ANDA$<>"N"THEN980
990 IFA$="J"THENRUN
995 I=I+1:IFI<(700-P-P1)/100THENPRINT"■ WIRKLICH? BEI"P+P1"PUNKTEN?":GOTO970
996 END
1000 REM ------------
1010 REM # ADDITION #
1020 REM ------------
1030 A$=MID$(STR$(A),2)+"+"+MID$(STR$(B),2)+"+"+MID$(STR$(C),2)+"= ?"
1040 RE=A+B+C
1050 RETURN
```

```
1100 REM ------------------------------
1110 REM # ADDITION & MULTIPLIKATION #
1120 REM ------------------------------
1130 A$=MID$(STR$(A),2)+"+"+MID$(STR$(B),2)+"*"+MID$(STR$(C),2)+"= ?"
1140 RE=B*C+A
1150 RETURN
1200 REM ------------------------------
1210 REM # ADDITION & MULTIPLIKATION #
1220 REM ------------------------------
1230 A$="("+MID$(STR$(A),2)+"+"+MID$(STR$(B),2)+")*"+MID$(STR$(C),2)+"= ?"
1240 RE=(A+B)*C
1250 RETURN
1300 REM ----------------------------
1310 REM # SUBTRAKTION & DIVISION #
1320 REM ----------------------------
1330 RE=A:A=RE*C+B
1340 A$="("+MID$(STR$(A),2)+"-"+MID$(STR$(B),2)+")/"+MID$(STR$(C),2)+"= ?"
1350 RETURN
2000 REM -----------
2010 REM ## SPIEL ##
2020 REM -----------
2030 PRINT"■";
2040 FORI=1TO24:PRINTA$(I,2)LEFT$(S1$,15);:NEXT
2050 GOSUB4110
2060 X0=22:Y0=48:X=12:Y=22:DX=0:DY=0
2070 TI$="000000"
2080 POKEV+0,(X0+8*X):POKEV+1,(Y0+8*Y)
2090 X1=X:Y1=Y
2100 GETA$
```

```
2110 IFA$="I"THENDX=0:DY=-.25
2120 IFA$="M"THENDX=0:DY=.25
2130 IFA$="J"THENDX=-.25:DY=0
2140 IFA$="K"THENDX=.25:DY=0
2150 IFTI$>"000030"THENRETURN
2160 PRINT"S[■■■■■■■■■■■■■■■■■■■■■■■■]"RIGHT$(TI$,2)
2170 IFDX=0ANDDY=0THEN2100
2180 X=X+DX:Y=Y+DY
2190 A=PEEK(1024+INT(Y+.4)*40+INT(X+.4))
2200 GOSUB4030
2210 IFP=188THEN2240
2220 GOTO2080
2230 REM ----------------------
2240 REM # KEIN FRESSEN MEHR #
2250 REM ----------------------
2260 A$=TI$:A1=0
2270 P1=P1+249-VAL(A$):P=0
2280 GOSUB4110
2290 A$="BONUS:"+STR$(60-VAL(A$)):S1=27:S2=22:GOSUB4200
2300 RETURN
4000 REM ---------------------
4010 REM ## UNTERPROGRAMME ##
4020 REM ---------------------
4030 IFA=46THEN4100
4040 IFA=32ORA=96THENRETURN
4050 XD=0:YD=0
4060 X=X1:Y=Y1
4070 RETURN
4100 POKE(1024+INT(Y+.4)*40+INT(X+.4)),32:P=P+1
```

```
4110 PRINT"SQ▮▮▮▮▮▮▮▮▮"P+P1"▮▮ "
4120 RETURN
4200 PRINT"S"LEFT$(S1$,S1)LEFT$(S2$,S2)A$;
4210 RETURN
4300 REM -----------------
4310 REM # ZAHLENEINGABE #
4320 REM -----------------
4330 R$="....":RE$=""
4340 A$=RIGHT$(R$+RE$,4):S1=29:S2=14:GOSUB4200
4350 GETA$:IFA$=""THEN4350
4360 IFA$=CHR$(13)THENRETURN
4370 IFA$=" "THEN4310
4380 IFA$<"0"ORA$>"9"THEN4350
4390 RE$=RE$+A$
4400 IFLEN(RE$)>4THEN4310
4410 A$=RIGHT$(R$+RE$,4):S1=29:S2=14:GOSUB4200
4420 GOTO4350
5000 REM --------------
5010 REM # RECHENFELD #
5020 REM --------------
5030 DATA"▮┌┬┬──────────┬┬┐
5040 DATA"||└──────────┘||
5050 DATA"|└──────────┘|
5060 DATA"|▮RECHENMAMPFER▮|
5070 DATA"|┌──────────┐|
5080 DATA"||┌──────────┐||
5090 DATA"├┴┴──────────┴┴┤
5100 DATA"|              |
5110 DATA"|              |
```

```
5120 DATA"|                    |
5130 DATA"|                    |
5140 DATA"|                    |
5150 DATA"|                    |
5160 DATA"|                    |
5170 DATA"|                    |
5180 DATA"|                    |
5190 DATA"|                    |
5200 DATA"|                    |
5210 DATA"|                    |
5220 DATA"|                    |
5230 DATA"|                    |
5240 DATA"|                    |
5250 DATA"|                    |
5260 DATA"└────────────────────┘
5270 REM -------------
5280 REM # SPIELFELD #
5290 REM -------------
5300 DATA"┌──────────────────────────────┐
5310 DATA"|PUNKTE:         ZEIT:       |
5320 DATA"└──────────────────────────────┘
5330 DATA"┌──────┬──────┬──────┬──────┐
5340 DATA"|■.....■|■.....■|■.....■|■.....■|
5350 DATA"|■.■┌──┐■|■..■┌──┐■|■..■┌──┐■|■.■|
5360 DATA"|■.■|1 3|■|■. .■└──┘■|■. .■|3 1|■.■|
5370 DATA"|■.■|4 2|■|■.........■|2 4|■.■|
5380 DATA"|■.■└──┘■|■.■┌──────┐■|■.■└──┘■|■.■|
5390 DATA"|■.....■└──────┘■.....■|
```

```
5400 DATA"|▒.■┌──┐▒.........■┌──┐▒.■|
5410 DATA"|▒.■|3+4|▒.■┌──────┐▒.■|3*4|▒.■|
5420 DATA"|▒.■└──┘▒.■|ΩRECHEN-■|▒.■└──┘▒.■|
5430 DATA"|▒.......■|        |▒.......■|
5440 DATA"|▒.■┌──┐▒.■|ΩMAMPFER■|▒.■┌──┐▒.■|
5450 DATA"|▒.■|4-3|▒.■└──────┘▒.■|4/3|▒.■|
5460 DATA"|▒.■└──┘▒.........■└──┘▒.■|
5470 DATA"|▒.......■┌──────┐▒.......■|
5480 DATA"|▒.■┌──┐▒.■└──────┘▒.■┌──┐▒.■|
5490 DATA"|▒.■|4 2|▒.........■|2 4|▒.■|
5500 DATA"|▒.■|1 3|▒. .■┌──┐▒. .■|3 1|▒.■|
5510 DATA"|▒.■└──┘▒...■└──┘▒...■└──┘▒.■|
5520 DATA"|▒.......■|▒.......■|▒.......■|
5530 DATA"└──────────┴──────────┴──────┘
5540 REM ------------------
5550 REM # RECHENMAMPFER #
5560 REM ------------------
5570 DATA 015,000,000,063,192,000,127,224,000,102,096,000,198,048,000
5580 DATA 255,240,000,207,048,000,102,096,000,112,224,000,063,192,000
5590 DATA 015,000,000,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
```

TEXTMON 64

Mit **TEXTMON** kannst du eigene Texte oder „Dateien“ herstellen und auf der Diskette speichern. Du kannst Diktate üben und vieles andere. Es ist ein Programm, das dich mit den Möglichkeiten der Textverarbeitung bekannt macht und mit dem du selbst experimentieren kannst.

Am Anfang erscheint das Inhaltsverzeichnis. Man nennt das ein Menü. Durch Drücken der entsprechenden Zahlen-Tasten kannst du auswählen, welche Aufgabe **TEXTMON** übernehmen soll.

Unter **Nummer 1** befindet sich ein kurzer Begrüßungstext.

Nummer 2 ermöglicht dir, Texte zu verfassen, elektronische „Kartei-Kästen“ anzulegen oder zum Beispiel ein Computer-Tagebuch zu führen. Beim Schreiben ist der Computer in diesem Programm etwas langsamer, als du es gewohnt bist. Während des Schreibens zählt der Computer die Zeilen und die Spalten (das ist die Zahl der Buchstaben und der Leerzeichen dazwischen). Ein Text kann nicht länger als 60 Zeilen sein.

Du kannst in deinen Texten alle **GROSSBUCHSTABEN, ZAHLEN** und **SATZZEICHEN** verwenden, allerdings *keine Anführungszeichen.*

Die Cursorsteuerung, das Radieren und das Überschreiben funktionieren so, wie du es gewohnt bist. Du kannst somit jederzeit Änderungen in deinem Text vornehmen.

```
| TEXTMONITOR          | ZEILE :   1
| ZEITPLAN             | SPALTE:   1

MITTWOCH, DER 16. OKTOBER 1985

 8.00  UHR   DEUTSCH
 8.45  UHR   MATHEMATIK
 9.30  UHR   PAUSE (KINOVERABREDUNG!!!)
10.00  UHR   DEUTSCH
10.45  UHR   HEIMAT- UND SACHKUNDE
11.30  UHR   PAUSE
11.45  UHR   MUSIK
12.30  UHR   SPORT
13.15  UHR   *** FREI ***

14.00  UHR   FUSSBALL
15.45  UHR   HAUSAUFGABEN
17.00  UHR   *** COMPUTERN ***

17.50  UHR   'DER ROSAROTE PANTHER'
```

Mit der **RUN/STOP-Taste** kommst du immer wieder ins Inhaltsverzeichnis (Menü) zurück.

Mit **Nummer 3** kannst du einen Text auf der Diskette abspeichern. Dazu muß dein Text allerdings einen Namen haben. Wenn du versuchst, einen Text ohne Namen abzuspeichern, schickt dich der Computer mit dem Hinweis darauf ins Menü zurück.

Nummer 6 gibt dir die Möglichkeit, den Textnamen festzulegen. Er darf allerdings höchstens 11 Buchstaben bzw. Zahlen umfassen.

Unter **Nummer 4** kannst du einen Text von der Diskette in deinen Computer laden. Wir haben drei Texte als Beispiele auf die Diskette geschrieben. Es ist genügend Platz auf der Diskette für deine eigenen Texte. Du kannst sie abspeichern, nachdem du ihnen einen Namen gegeben hast.

```
      TEXT VON DISK LADEN

DIESE TEXTE STEHEN AUF DER EINGELEGTEN
DISKETTE:
  1) APFELKUCHEN          2) IM GRASE
  3) ZEITPLAN

GEBE BITTE DIE ZAHL DES GEWUENSCHTEN
TEXTES EIN.

DER TEXT >APFELKUCHEN< WIRD GELADEN.

EINGELADENE ZEILE: 39
```

Der Computer zeigt dir zuerst an, welche Texte sich auf der Diskette befinden. Dann mußt du nur noch die gewünschte Zahl eingeben und mit RE-TURN bestätigen.

Während der Computer lädt, zählt er die Zeilen des Textes. Auch hier kannst du mit RUN/STOP ins Menü zurückkehren.

Mit **Nummer 5** kannst du einen Text, den du nicht mehr brauchen kannst, von der Diskette löschen. Dazu brauchst du nur die Frage des Computers mit J für Ja zu beantworten. Falls du jedoch aus Versehen in diesem Programmteil geraten bist, kannst du mit der Antwort N für Nein verhindern, daß ein Text gelöscht wird.

Die **Nummer 7** gibt dir die Möglichkeit, ein Diktat zu üben, und zwar Zeile für Zeile. Ebenso wie ein deutsches Diktat ist auch ein fremdsprachiges möglich, solange man sich auf Großbuchstaben beschränken kann.

Zum Diktat wird immer der Text verwendet, der gerade geladen ist. Falls du also ein neues Diktat üben willst, mußt du zuerst den Text dazu schreiben oder einen bestimmten Text von der Diskette laden.

Der Computer fragt, wie lange eine Zeile auf dem Bildschirm zu sehen sein soll, bevor sie verschwindet und du sie aus dem Gedächtnis nachschreiben sollst. Die Wahl 1 Sekunde ist vor allem für einzelne Worte (Vokabeln) und für Gedächtnisübungen be-

stimmt, die Wahl 2 Minuten für besonders schwierige Sätze oder Formeln.

```
        TEXT WIRD DIKTIERT

WIE LANGE SOLL EINE TEXTZEILE AUF DEM
BILDSCHIRM ERSCHEINEN?

                1 SEKUNDEN
                2 SEKUNDEN
                4 SEKUNDEN
                8 SEKUNDEN
               16 SEKUNDEN
               32 SEKUNDEN
               64 SEKUNDEN
                2 MINUTEN

GEBE DIE GEWUENSCHTE ZAHL EIN !
```

Mit RUN/STOP kannst du auch ein Diktat unterbrechen, um ins Menü zu kommmen (zum Beispiel, um einen anderen Text zu laden).

Mit ”0“ beendest du das ganze Programm.

```
100 REM ---------------------------
110 REM >  T E X T M O N    6 4  <
120 REM >                        <
130 REM >    PROGRAMMIERT VON    <
140 REM >    PETER  FRITZSCHE    <
150 REM >                        <
160 REM >    IM  AUFTRAG  VOM    <
170 REM > FRANZ SCHNEIDER VERLAG <
180 REM ---------------------------
190 REM
200 CLOSE1:CLOSE138
210 POKE53269,0:POKE53271,0:POKE53277,0
220 POKE788,52:REM  >RUN/STOP< AUS
230 POKE792,193:REM >RESTORE < AUS
240 POKE53280,0:POKE53281,7:PRINT"■■■■■■■■■■■■■■■■■■■■BITTE EINEN MOMENT WARTEN!"
250 SP$="                                                ":V=53248
260 DIMA$(60),T$(40)
270 GOSUB11000
280 DATA255,192,,128,64,,128,64,,128,64,,128,64,,128,64,,128,64,,128,64,,128,64,
290 DATA255,192,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
300 FORI=0TO62:READJ:POKE832+I,J:NEXT
310 POKE2040,13:POKEV+21,1:POKEV+39,0
320 OPEN1,8,0,"$0":GET#1,A$,B$:T=0
330 GET#1,A$,B$:I=I+1
340 GET#1,A$,B$:C=0
350 IFA$<>""THENC=ASC(A$)
360 IFB$<>""THENC=C+ASC(B$)*256
370 GET#1,B$:IFST<>0THEN450
380 C$="":IFB$<>CHR$(34)THEN370
```

```
390 GET#1,B$:IFB$<>CHR$(34)THENC$=C$+B$:GOTO390
400 IFLEFT$(C$,2)="T-"THENT=T+1:T$(T)=MID$(C$,3)
410 GET#1,B$:IFB$=CHR$(32)THEN410
420 PRINTTAB(25);:C$=""
430 C$=C$+B$:GET#1,B$:IFB$<>""THEN430
440 IFST=0THEN330
450 CLOSE1:IFT<2THEN500
460 FORI=1TOT-1:FORJ=I+1TOT
470 IFT$(I)>T$(J)THENQ$=T$(I):T$(I)=T$(J):T$(J)=Q$
480 NEXTJ,I
500 REM -----------------
510 REM > HAUPTPROGRAMM <
520 REM -----------------
530 POKEV+1,0
540 PRINT"┌--------------------------------------------";
550 PRINT"|                                                  |";
560 PRINT"|                                            |";
570 PRINT"|                                          |";
580 PRINT"└-----------------------------------------"
590 PRINT" R>_1_<■   TEXTMON EINFUEHRUNG
600 PRINT" R>_2_<■   TEXTMONITOR
610 PRINT" R>_3_<■   TEXT ABSPEICHERN
620 PRINT" R>_4_<■   TEXT LADEN
630 PRINT" R>_5_<■   TEXT LOESCHEN
640 PRINT" R>_6_<■   TEXTNAMEN VERGEBEN
650 PRINT" R>_7_<■   TEXT DIKTIEREN
660 PRINT"
670 PRINT"
680 PRINT" R> 0 <■   PROGRAMM ABBRECHEN
```

```
690 PRINT"■■  DRUECKE BITTE GEWUENSCHTE TASTE !"
700 GETQ$:IFQ$<"0"ORQ$>"9"THEN700
710 Q=VAL(Q$)
720 ON Q GOTO 1000,2000,3000,4000,5000,6000,7000,500,500
730 IFSP=1THEN12000
740 POKE788,49:REM >RUN/STOP< EIN
750 POKE792,71:REM >RESTORE < EIN
760 PRINT"■":END
1000 REM ---------------
1010 REM > EINFUEHRUNG <
1020 REM ---------------
1030 PRINT"■┌──────────────────────────────────────────┐";
1040 PRINT"│    PROGRAMMEINFUEHRUNG      SEITE 1    │";
1050 PRINT"└──────────────────────────────────────────┘"
1060 PRINT" DIESES PROGRAMM IST EINE EINFUEHRUNG "
1070 PRINT" IN DIE TEXTVERARBEITUNG.    ":PRINT
1080 PRINT" MAN KANN DAMIT AUCH DIKTATE UEBEN."
1090 PRINT
1100 PRINT" EINE AUSFUEHRLICHE ANLEITUNG DAZU "
1110 PRINT" FINDEST DU IM BUCH 'COMPUTER KINDER-    LEICHT BAND II'."
1120 GOSUB10000
1130 GOTO500
2000 REM ---------------
2010 REM > TEXTMONITOR <
2020 REM ---------------
2030 IM=1:X=1:Y=1:SP=1:PRINT"■"
2040 X$=RIGHT$(SP$+STR$(X),3):Y$=RIGHT$(SP$+STR$(Y),3)
```

```
2050 PRINT"■┌──────┬──────────────────────────────────────┐";
2060 PRINT"| TEXTMONITOR | ZEILE :"Y$"                    |";
2070 PRINT"| "LEFT$(NA$+SP$,11)" | SPALTE:"X$"                 |";
2080 PRINT"└──────┴──────────────────────────────────────┘";
2090 FORI=IMTOIM+19:PRINTA$(I);:NEXT
2100 Xi=X*8+15:Yi=(Y-IM)*8+80
2110 POKEV+16,INT(X1/255):POKEV,X1-255*INT(X1/255):POKEV+1,Y1
2120 GETQ$:IFQ$=""THEN2120
2130 IFQ$>CHR$(34)ANDQ$<"↑"ORQ$="!"ORQ$=" "THENGOSUB2290:GOTO2040
2140 IFQ$="■"ORQ$="▚"THENX=1:Y=IM:GOTO2100
2150 IFQ$=CHR$(20)ANDX>1THENGOSUB2360:GOTO2040
2160 IFQ$=CHR$(148)ANDX<39THENGOSUB2380:GOTO2040
2170 IFQ$="▒"THENGOSUB2260:GOTO2040
2180 IFQ$="▌"THENGOSUB2240:GOTO2040
2190 IFQ$="▐"THENGOSUB2310:GOTO2040
2200 IFQ$="▀"THENGOSUB2330:GOTO2040
2210 IFQ$=CHR$(13)ORQ$=CHR$(141)THENGOSUB2320:GOTO2040
2220 IFQ$="▄"THEN500
2230 GOTO2120
2240 X=X-1:IFX>0THENRETURN
2250 X=40
2260 Y=Y-1:IFY<1THENY=1:X=1:RETURN
2270 IFY<IMTHENIM=IM-5
2280 RETURN
2290 A$(Y)=A$(Y)+"*"
2300 A$(Y)=LEFT$(LEFT$(A$(Y),X-1)+Q$+MID$(A$(Y),X+1),40)
2310 X=X+1:IFX<41THENRETURN
2320 X=1
2330 Y=Y+1:IFY>60THENY=60:RETURN
```

```
2340 IFY-IM>19THENIM=IM+5
2350 RETURN
2360 A$(Y)="*"+A$(Y)
2370 A$(Y)=MID$(LEFT$(A$(Y),X-1)+MID$(A$(Y),X+1)+" ",2):X=X-1:RETURN
2380 A$(Y)=LEFT$(LEFT$(A$(Y),X-1)+" "+MID$(A$(Y),X),40):RETURN
2390 GOSUB10000:GOTO500
3000 REM ---------------------
3010 REM > TEXT ABSPEICHERN <
3020 REM ---------------------
3030 PRINT"□┌──────────────────────────────────────────┐";
3040 PRINT"│    TEXT  ABSPEICHERN                     │";
3050 PRINT"└──────────────────────────────────────────┘"
3060 IFNA$=""ORLEFT$(NA$,1)="?"THEN3240
3070 PRINT" ACHTUNG! TEXTE, DIE DEN NAMEN"
3080 PRINT" "NA$" BESITZEN, WERDEN UEBER-"
3090 PRINT" SCHRIEBEN."
3100 PRINT"□ SOLL DER TEXT ABGESPEICHERT WERDEN?"
3110 PRINTTAB(15)"(J=JA/N=NEIN)"
3120 GETQ$:IFQ$<>"J"ANDQ$<>"N"THEN3120
3130 IFQ$="N"THEN500
3140 PRINT"□ DER TEXT WIRD ABGESPEICHERT."
3150 OPEN138,8,2,"@:T-"+NA$+",S,W"
3160 FORI=1TO60:PRINT"□ ABGESPEICHERTE ZEILE:"I"□□"
3170 A$="":FORJ=1TO40
3180 A$=A$+CHR$(24+ASC(MID$(A$(I),J,1))):NEXT
3190 PRINT#138,">"+LEFT$(A$+SP$,40)+"<";CHR$(13);
3200 NEXT:CLOSE138:SP=0
3210 J=0:FORI=1TOT:IFT$(I)=NA$THENJ=13
3220 NEXT:IFJ<>13THENT=T+1:T$(T)=NA$
```

```
3230 GOTO450
3240 PRINT" ES WURDE FUER DEN TEXT NOCH KEIN NAME   @VERGEBEN"
3250 GOSUB10000:GOTO500
4000 REM ---------------
4010 REM > TEXT LADEN <
4020 REM ---------------
4030 PRINT"@┌──────────────────────────────────────────┐";
4040 PRINT"|     TEXT VON DISK LADEN                  |";
4050 PRINT"└──────────────────────────────────────────┘"
4060 IFT<1THENPRINT"@ ES SIND KEINE TEXTE AUF DER DISK":GOSUB10000:GOTO500
4070 PRINT" DIESE TEXTE STEHEN AUF DER EINGELEGTEN"
4080 PRINT" DISKETTE:"
4090 FORI=1TOT
4100 PRINT" "RIGHT$(SP$+STR$(I),2)") "LEFT$(T$(I)+SP$,15);
4110 NEXT:PRINT
4120 PRINT"@ GEBE BITTE DIE ZAHL DES GEWUENSCHTEN"
4130 PRINT" TEXTES EIN.@"
4140 C$="  "
4150 PRINT" DIE ZAHL: "C$
4160 GETQ$:IFQ$=""THEN4160
4170 IFQ$>"/"ANDQ$<":"THENC$=RIGHT$(C$+Q$,2):PRINT"□";:GOTO4150
4175 IFQ$="@"THEN500
4180 IFQ$<>CHR$(13)ANDQ$<>CHR$(141)THEN4160
4190 I=VAL(C$):IFI<1ORI>TTHEN4160
4200 NA$=T$(I):PRINT"□ DER TEXT >"NA$"< WIRD GELADEN."
4210 OPEN138,8,4,"T-"+T$(I)+",S,R"
4220 FORI=1TO60:PRINT"@ EINGELADENE ZEILE:"I"□□"
4230 INPUT#138,Q$:A$=LEFT$(MID$(Q$,2,40)+SP$,40)
4240 A$(I)="":FORJ=1TO40:A$(I)=A$(I)+CHR$(ASC(MID$(A$,J,1))-24)
```

```
4250 NEXTJ,I :CLOSE138:GOTO2000
5000 REM ------------------
5010 REM > TEXT LOESCHEN <
5020 REM ------------------
5030 PRINT"■┌──────────────────────────────────────────┐";
5040 PRINT"|    TEXT  LOESCHEN                        |";
5050 PRINT"└──────────────────────────────────────────┘"
5060 PRINTTAB(17)"■■SOLL"
5070 PRINT"■           DER TEXT "NA$
5080 PRINT"■ GELOESCHT WERDEN? (J=JA,N=NEIN)"
5090 GETQ$:IFQ$<>"J"ANDQ$<>"N"THEN5090
5100 IFQ$="N"THEN500
5150 SP=0:GOSUB11000:GOTO450
6000 REM ----------------------
6010 REM > TEXTNAMEN     <
6020 REM ----------------------
6030 NA$="":PRINT"■"
6040 PRINT"■┌──────────────────────────────────────────┐";
6050 PRINT"|    TEXTNAMEN  VERGEBEN                   |";
6060 PRINT"└──────────────────────────────────────────┘"
6070 PRINT" DER NAME DARF AUS MAX. 11 BUCHSTABEN    BESTEHEN."
6080 PRINT"■ DER TEXTNAME LAUTET: "NA$" "
6090 PRINT"■■■■■ WENN DU FERTIG BIST, DRUECKE RUN/STOP!"
6100 GETQ$:IFQ$=""THEN6100
6110 IFQ$>="A"ANDQ$<="Z"ORQ$=" "THENNA$=NA$+Q$
6120 IFQ$=CHR$(20)ANDLEN(NA$)<2THENNA$=""
6130 IFQ$=CHR$(20)ANDLEN(NA$)>1THENNA$=LEFT$(NA$,LEN(NA$)-1)
6140 IFQ$="■"ORQ$=CHR$(13)THEN500
6150 NA$=LEFT$(NA$,11):GOTO6040
```

```
7000 REM ------------------
7010 REM > TEXT DIKTIEREN <
7020 REM --------------
7030 PRINT"■┌──────────────────────────────────────────┐";
7040 PRINT"│    TEXT WIRD DIKTIERT                    │";
7050 PRINT"└──────────────────────────────────────────┘"
7060 PRINT" WIE LANGE SOLL EINE TEXTZEILE AUF DEM  ";
7070 PRINT" BILDSCHIRM ERSCHEINEN?"
7080 PRINT"■ ■>_1_<■  1 SEKUNDE
7090 PRINT" ■>_2_<■  2 SEKUNDEN
7100 PRINT" ■>_3_<■  4 SEKUNDEN
7110 PRINT" ■>_4_<■  8 SEKUNDEN
7120 PRINT" ■>_5_<■ 16 SEKUNDEN
7130 PRINT" ■>_6_<■ 32 SEKUNDEN
7140 PRINT" ■>_7_<■ 64 SEKUNDEN
7150 PRINT" ■>_8_<■  2 MINUTEN
7160 PRINT"■ GEBE DIE GEWUENSCHTE ZAHL EIN !"
7170 GETQ$:IFQ$<"1"ORQ$>"8"THEN7170
7180 Q=2↑(VAL(Q$)-1)*819
7190 F=0:FORI=1TO60:IFA$(I)=SP$THEN7550
7200 PRINT"■┌──────────────────────────────────────────┐";
7210 PRINT"│    TEXT WIRD DIKTIERT                    │";
7220 PRINT"└──────────────────────────────────────────┘"
7230 PRINT" NUM WIRD DER TEXT ■"NA$"■ DIKTIERT"
7240 PRINT"■ ZEILE:"I
7250 PRINT"■ DIE ZEILE LAUTET:■"
7260 PRINTA$(I)"■";
7270 FORJ=0TOQ:NEXT
7280 PRINT"□"SP$
```

```
7290 A$(0)=SP$:X=1:Y=0
7300 PRINT"□□□ NUN BIST DU DRAN MIT DEM EINGEBEN      "A$(0)
7310 X1=X*8+15:Y1=(Y+5)*8+80
7320 POKEV+16,INT(X1/255):POKEV,X1-255*INT(X1/255):POKEV+1,Y1
7330 GETQ$:IFQ$=""THEN7330
7340 IFQ$>CHR$(34)ANDQ$<"↑"ORQ$="!"ORQ$=" "THENGOSUB2290:Y=0:GOTO7300
7350 IFQ$=CHR$(20)ANDX>1THENGOSUB2360:GOTO7300
7360 IFQ$=CHR$(148)ANDX<39THENGOSUB2380:GOTO7300
7370 IFQ$="▌"THENGOSUB2240:Y=0:GOTO7300
7380 IFQ$="▐"THENGOSUB2310:Y=0:GOTO7300
7390 IFQ$<>CHR$(13)ANDQ$<>CHR$(141)ANDQ$<>"■"THEN7330
7400 POKEV+16,0:POKEV,0
7410 IFQ$="■"THEN500
7420 A$(0)=LEFT$(A$(0)+SP$,40)
7430 IFA$(0)=A$(I)THENPRINT" DEINE EINGABE STIMMT!":FORJ=0TO2000:NEXT:GOTO7550
7440 FORJ=40TO1STEP-1
7450 IFMID$(A$(0),J,1)<>MID$(A$(I),J,1)THENX=J
7460 NEXT
7470 X1=X*8+15:Y1=(Y+5)*8+80
7480 POKEV+16,INT(X1/255):POKEV,X1-255*INT(X1/255):POKEV+1,Y1
7490 PRINT" DU HAST EINEN FEHLER BZW. MEHRERE FEH-":F=1
7500 PRINT" LER BEI DEINER EINGABE GEMACHT."
7510 PRINT"■ DIE ZEILE LAUTETE:"
7520 PRINTA$(I);
7530 PRINT"■ VERSUCH ES NOCH EINMAL!"
7540 FORJ=0TO8000:NEXT:POKEV,0:POKEV+16,0:GOTO7200
7550 NEXTI
7560 IFF=0THENPRINT" DU HAST DEN TEXT OHNE FEHLER NACHGE-    TIPPT."
7570 IFF=1THENPRINT" DU HAST DEN TEXT MIT FEHLERN NACHGE-    TIPPT."
```

```
7580 IFF=1THENPRINT" WILLST DU ES NOCH MAL VERSUCHEN?"
7590 IFF=1THENPRINT" DANN DRUECKE IM HAUPTMENUE >7< !"
7600 FORI=0TO8000:NEXT:GOTO500
10000 REM ------------------
10010 REM > WEITERMACHEN ? <
10020 REM ------------------
10030 PRINT"QQQ WENN DU SOWEIT BIST,[■■■■■■■■DANN DRUECKE EINE TASTE!";
10040 GETQ$:IFQ$=""THEN10040
10050 RETURN
11000 FORI=1TO60:A$(I)=SP$:NEXT
11010 NA$="???????????":RETURN
12000 REM ----------------------------
12010 REM > TEXT NICHT ABGESPEICHERT <
12020 REM ----------------------------
12030 PRINT"QQQQQQQQQQ    IST DER TEXT ABGESPEICHERT ?"
12040 PRINT"Q SOLL PROGRAMM WIRKLICH BEENDET WERDEN?"
12050 PRINT"Q"TAB(10)"(J=JA,N=NEIN)"
12060 GETQ$:IFQ$=""THEN12060
12070 IFQ$="J"THEN740
12080 IFQ$="N"THEN500
12090 GOTO12060

READY.
```

```
100 :        ----------------
101 :         STEUERZEICHEN
102 :        ----------------
103 :
104 :     1) CURSORSTEUERTASTEN
105 :    "■" CRSR              CURSOR RECHTS
106 :    "■" SHIFT CRSR        CURSOR LINKS
107 :    "■" CRSR              CURSOR RUNTER
108 :    "■" SHIFT CRSR        CURSOR HOCH
109 :    "■" CLRHOME           CURSOR INS LINKE ECK
110 :    "■" SHIFT CLRHOME     BILDSCHIRM LOESCHEN
111 :
112 :     2) FUNKTIONSTASTEN
113 :    "■" F1                FUNKTIONSTASTE 1
114 :    "■" SHIFT F1          FUNKTIONSTASTE 2
115 :    "■" F3                FUNKTIONSTASTE 3
116 :    "■" SHIFT F3          FUNKTIONSTASTE 4
117 :    "■" F5                FUNKTIONSTASTE 5
118 :    "■" SHIFT F5          FUNKTIONSTASTE 6
119 :    "■" F7                FUNKTIONSTASTE 7
120 :    "■" SHIFT F7          FUNKTIONSTASTE 8
121 :
122 :     3) FARBEN
123 :    "■" CTRL 1            SCHWARZ
124 :    "■" CTRL 2            WEISS
125 :    "■" CTRL 3            ROT
126 :    "■" CTRL 4            TUERKIS
127 :    "■" CTRL 5            VIOLETT
128 :    "■" CTRL 6            GRUEN
129 :    "■" CTRL 7            BLAU
130 :    "■" CTRL 8            GELB
131 :    "■" C= 1              ORANGE
132 :    "■" C= 2              BRAUN
133 :    "■" C= 3              HELLROT
134 :    "■" C= 4              GRAU 1
135 :    "■" C= 5              GRAU 2
136 :    "■" C= 6              HELLGRUEN
137 :    "■" C= 7              HELLBLAU
138 :    "■" C= 8              GRAU 3
139 :
140 :     4) SONSTIGE STEUERZEICHEN
141 :    "■" CTRL 9                 REVERSSCHRIFT AN
142 :    "■" CTRL 0                 REVERSSCHRIFT AUS

READY.
```

Listings selbstgeschriebener Programme:

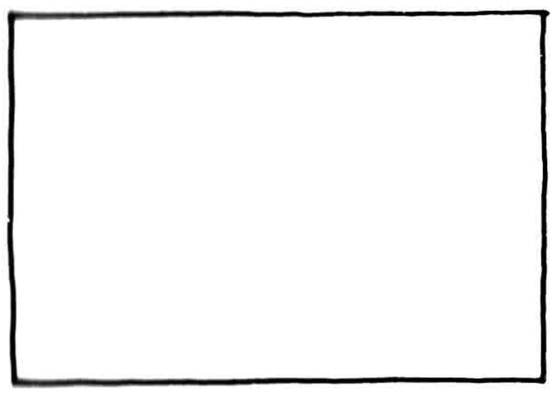
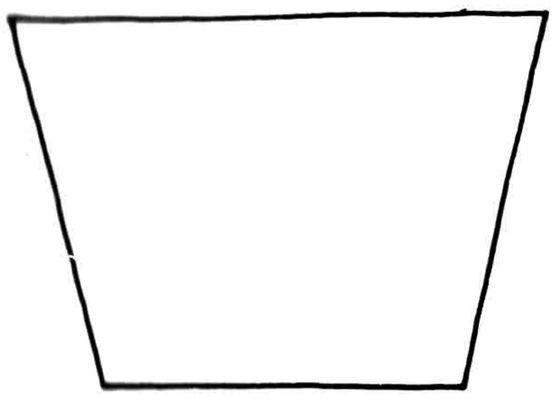
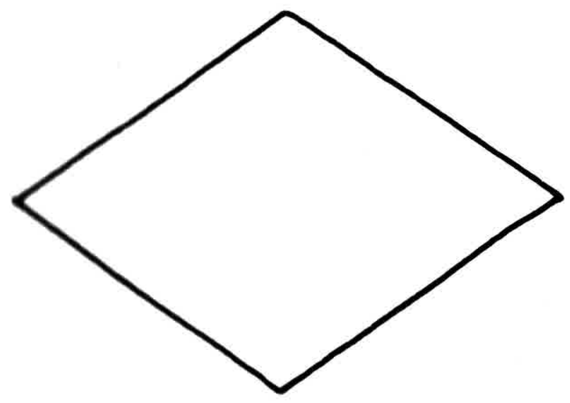
Zeilennummer	Anweisung

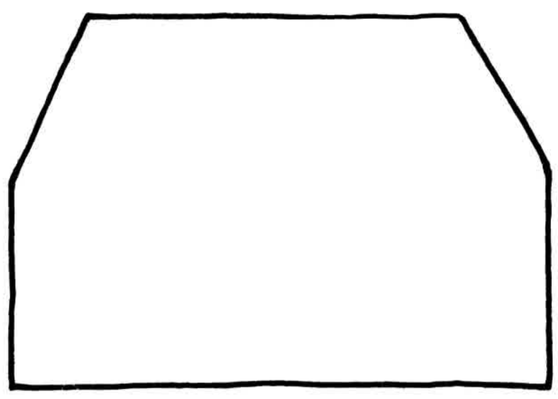
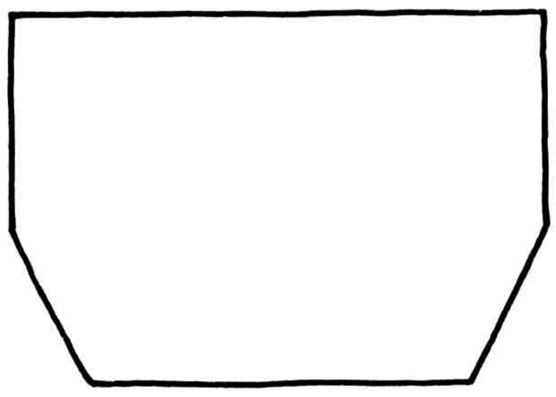
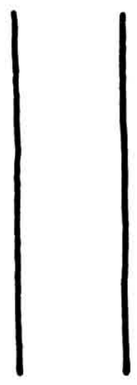
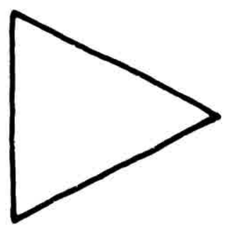
Hinweis an unsere Leser: Vorschläge zur Verbesserung und Kritik nehmen wir gerne entgegegen – Lob auch!

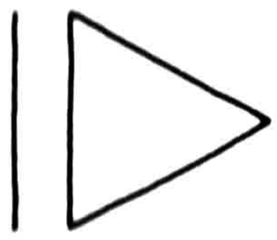
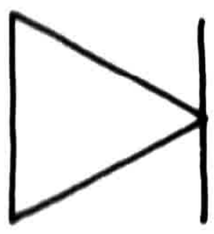
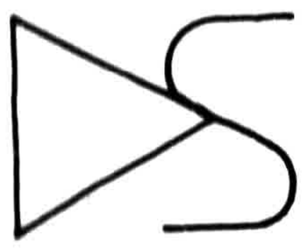
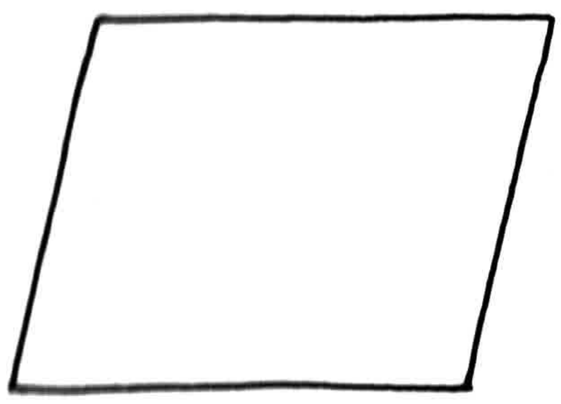
Unsere Adresse: SchneiderBuch
Redaktion
8000 München 100

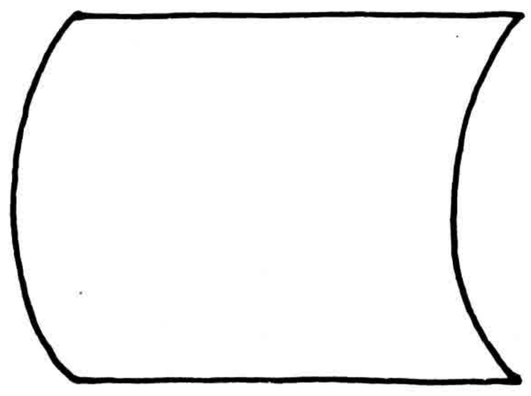
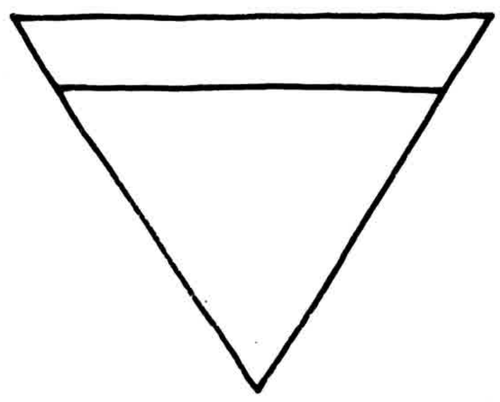
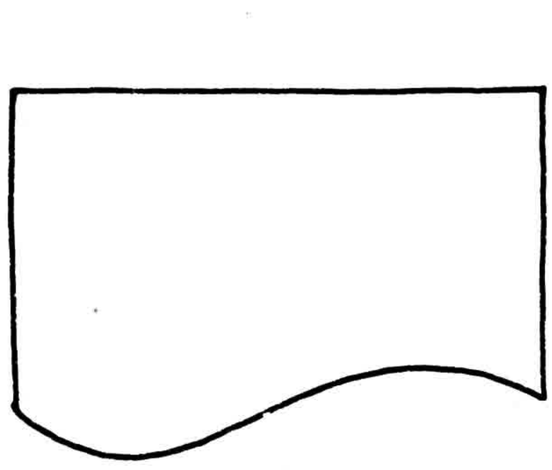
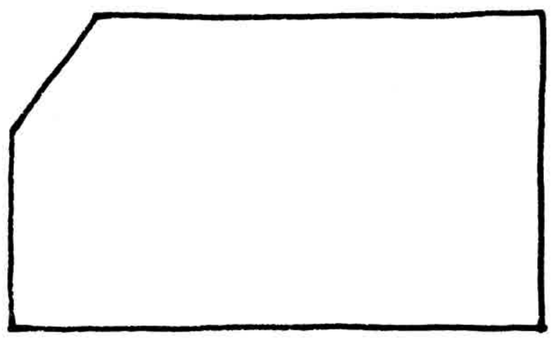
FÜR ALLE KÖNNER:

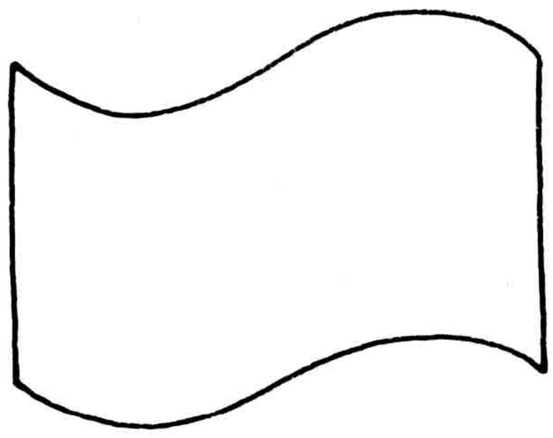
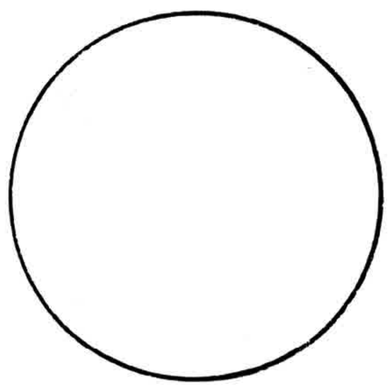
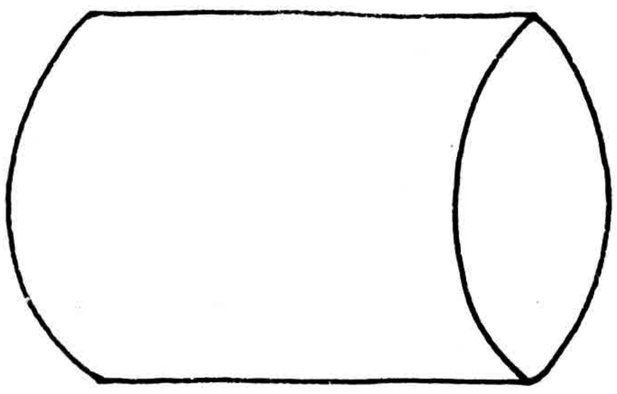
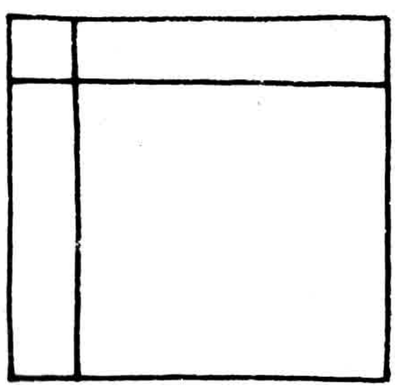
Sinnbilder zum Entwerfen von Programmen

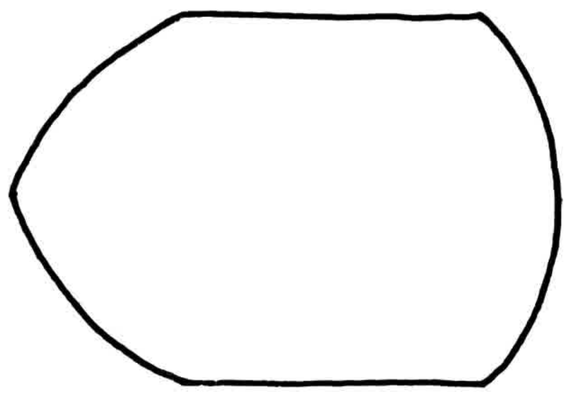
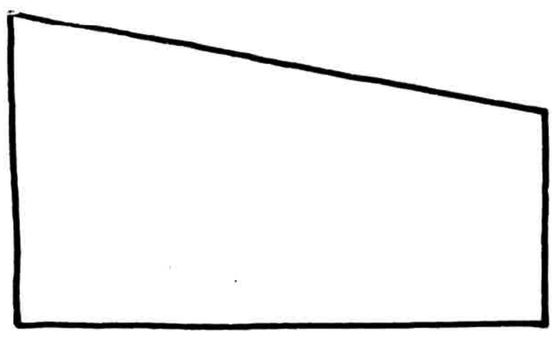
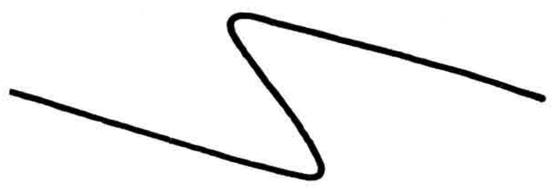
Sinnbild	Benennung und Bemerkung
Verarbeitungen/Verarbeitungseinheiten	
	Verarbeitung, allgemein (process) (einschließlich Ein-/Ausgabe) Verarbeitungseinheit, allgemein (processing unit)
	Manuelle Verarbeitung (manual operation) (einschließlich Ein-/Ausgabe) Manuelle Verarbeitungsstelle (point of manual operation) (z. B. Fachabteilung, Bediener)
	Verzweigung[1]) (decision) Auswahleinheit (selection unit) (z. B. Schalter)
[1]) Begriff siehe DIN 44 300	

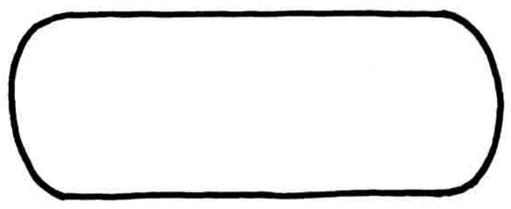
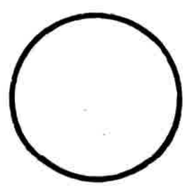
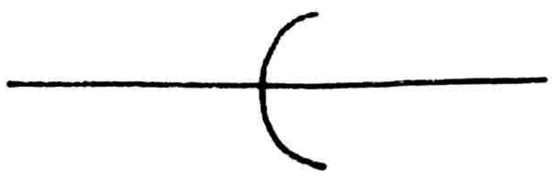
Sinnbild	Benennung und Bemerkung
	Schleifenbegrenzung (loop limit) Anfang
	Ende
	Synchronisierung paralleler Verarbeitungen (synchronization of parallel processes) **Hinweis:** Die Verbindungslinien müssen senkrecht zu den Linien des Sinnbildes geführt werden. Synchronisiereinheit[2]) (synchronization unit)
	Sprung mit Rückkehr (branch with return)
[2]) Begriff siehe DIN 44 302	

Sinnbild	Benennung und Bemerkung
	Sprung ohne Rückkehr (branch without return)
	Unterbrechung einer anderen Verarbeitung (interrupt of a different process)
	Steuerung der Verarbeitungsfolge von außen (external control of process) Anmerkung: Die von außen steuernde Verarbeitung ist nicht in der Darstellung enthalten (z. B. Steuerung durch ein Betriebssystem)
Daten[1])	
	Daten, allgemein (data) Datenträgereinheit, allgemein (data medium unit)
[1]) Begriff siehe DIN 44 300	

Sinnbild	Benennung und Bemerkung
	Maschinell zu verarbeitende Daten (data to be processed by machine) Datenträgereinheit für maschinell verarbeitbare Daten (input-output unit)
	Manuell zu verarbeitende Daten (data to be processed by manual operation) Manuelle Ablage (offline storage) (z. B. Ziehkartei, Archiv)
	Daten auf Schriftstück (data on document) (z. B. auf Belegen, Mikrofilm) Ein-/Ausgabeeinheit[1]) für Schriftstücke (input/output unit for documents) (z. B. Belegleser, Drucker, COM-Rekorder)
	Daten auf Karte (data on card) (z. B. Lochkarte, Magnetkarte) Lochkarteneinheit (punched card unit)
[1]) Begriff siehe DIN 44 300	

Sinnbild	Benennung und Bemerkung
	Daten auf Lochstreifen[3]) (data on punched tape) Lochstreifeneinheit (punched tape unit)
	Daten auf Speicher[1]) mit **nur** sequentiellem Zugriff (data on sequential access storage) Datenträgereinheit mit **nur** sequentiellem Zugriff (sequential access storage unit)
	Daten auf Speicher mit auch direktem Zugriff (data on direct access storage) Datenträgereinheit mit auch direktem Zugriff (direct access storage unit)
	Daten im Zentralspeicher (data in internal storage) Zentralspeicher[1]) (processor storage)
[3]) Begriff siehe DIN 66 218	

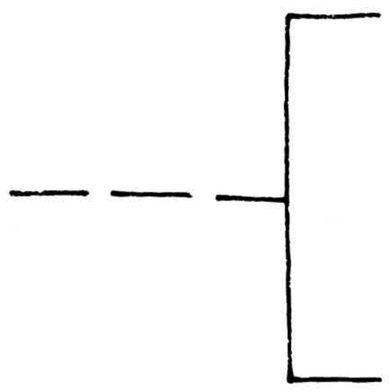
Sinnbild	Benennung und Bemerkung
	Maschinell erzeugte optische oder akustische Daten (displayed or acoustical output data) Optische oder akustische Ausgabeeinheit (display or acoustical output unit)
	Manuelle optische oder akustische Eingabedaten (manual input data) Eingabeeinheit (manual input unit)
Verbindungen[4])	
	Verbindung (line): Verarbeitungsfolge Zugriffsmöglichkeit Zugriffsweg Über-/Unterordnung Zusammenfassung/Unterteilung
	Verbindung zur Darstellung der Datenübertragung[2]) (communication link) Datenübertragungsweg (communication link)

Sinnbild	Benennung und Bemerkung
Darstellungshilfen	
	Grenzstelle (zur Umwelt) (terminator) (z. B. Beginn oder Ende einer Folge, Herkunft oder Verbleib von Daten)
	Verbindungsstelle (connector) Eine Verbindung kann durch eine Verbindungsstelle unterbrochen und an anderer Stelle derselben Darstellung mit einer Verbindungsstelle mit gleicher Innenbeschriftung fortgesetzt werden.
	Verfeinerung[5]) (refinement)

[2]) Begriff siehe DIN 44 302

[4]) Bei den „Verbindungen“ gilt die Vorzugsrichtung von links nach rechts bzw. von oben nach unten, Abweichungen sind durch Pfeilspitzen zu kennzeichnen. Die Form der Pfeilspitzen ist beliebig.
Im Konfigurationsplan (KP) gelten Verbindungen als beidseitig gerichtet (Ein- und Ausgabe). Spezielle Verbindungen (nur Eingabe oder nur Ausgabe) können durch Pfeilspitzen hervorgehoben werden.

[5]) Mit dem Sinnbild „Verfeinerung“ besteht die Möglichkeit, auf demselben Blatt in der gleichen Darstellungsart den Inhalt eines anderen Sinnbildes oder einer Teildarstellung in Richtung der Verbindung zu verfeinern. Wird eine Teildarstellung verfeinert, kann diese zur Verdeutlichung des verfeinerten Bereichs mit einer durchbrochenen Linie umrahmt werden.

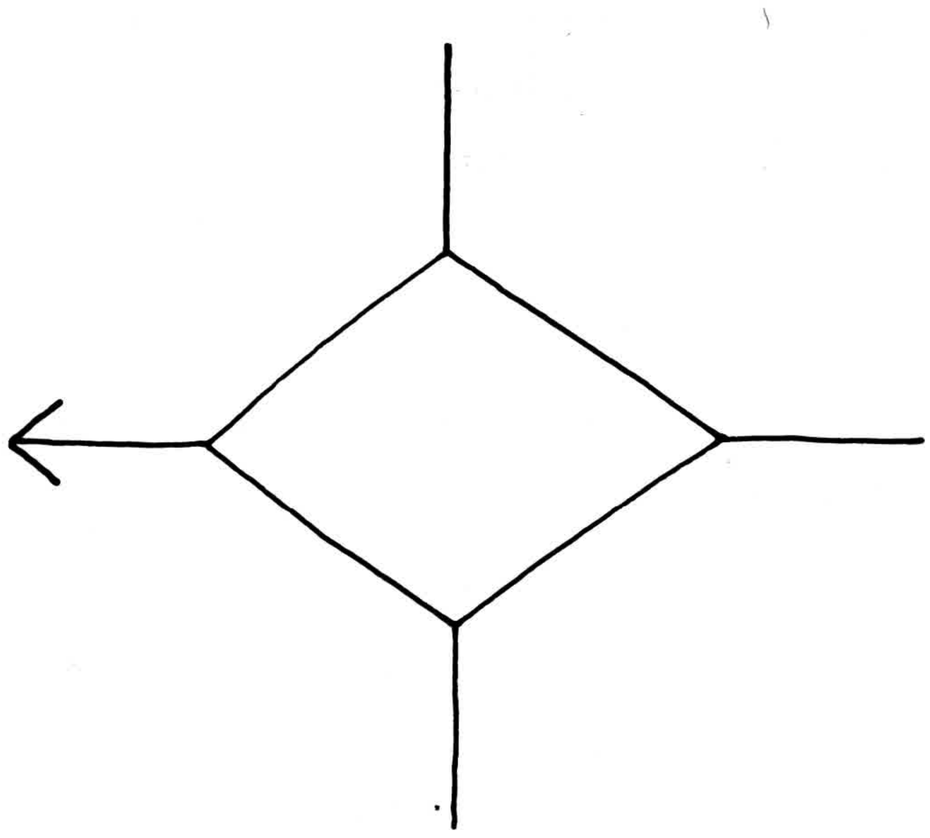
Sinnbild	Benennung und Bemerkung
	Bemerkung (annotation) Mit diesem Sinnbild kann erläuternder Text jedem anderen Sinnbild dieser Norm zugeordnet werden. Die durchbrochene Linie zum erläuterten Sinnbild darf durch eine Vollinie ersetzt werden.

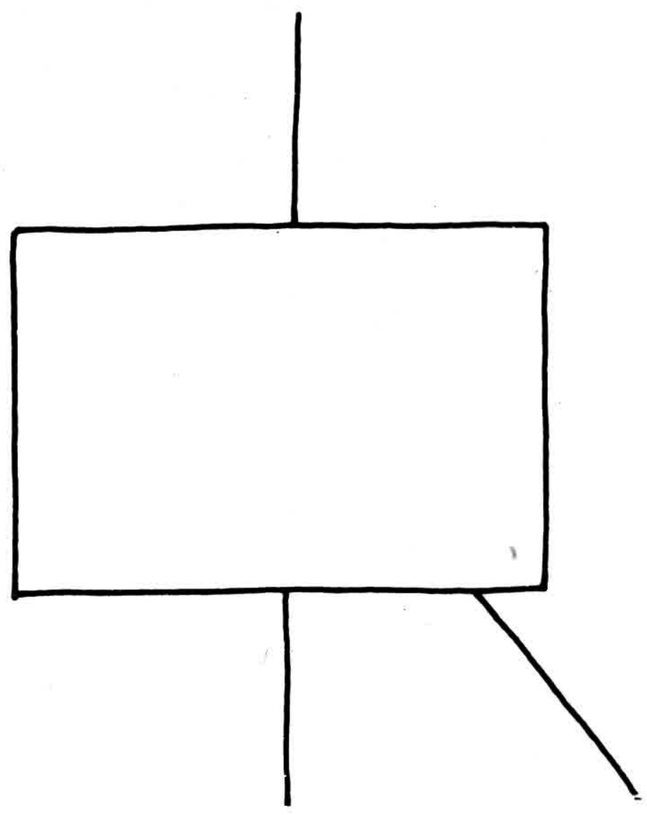
Darstellungshinweise

Anordnung

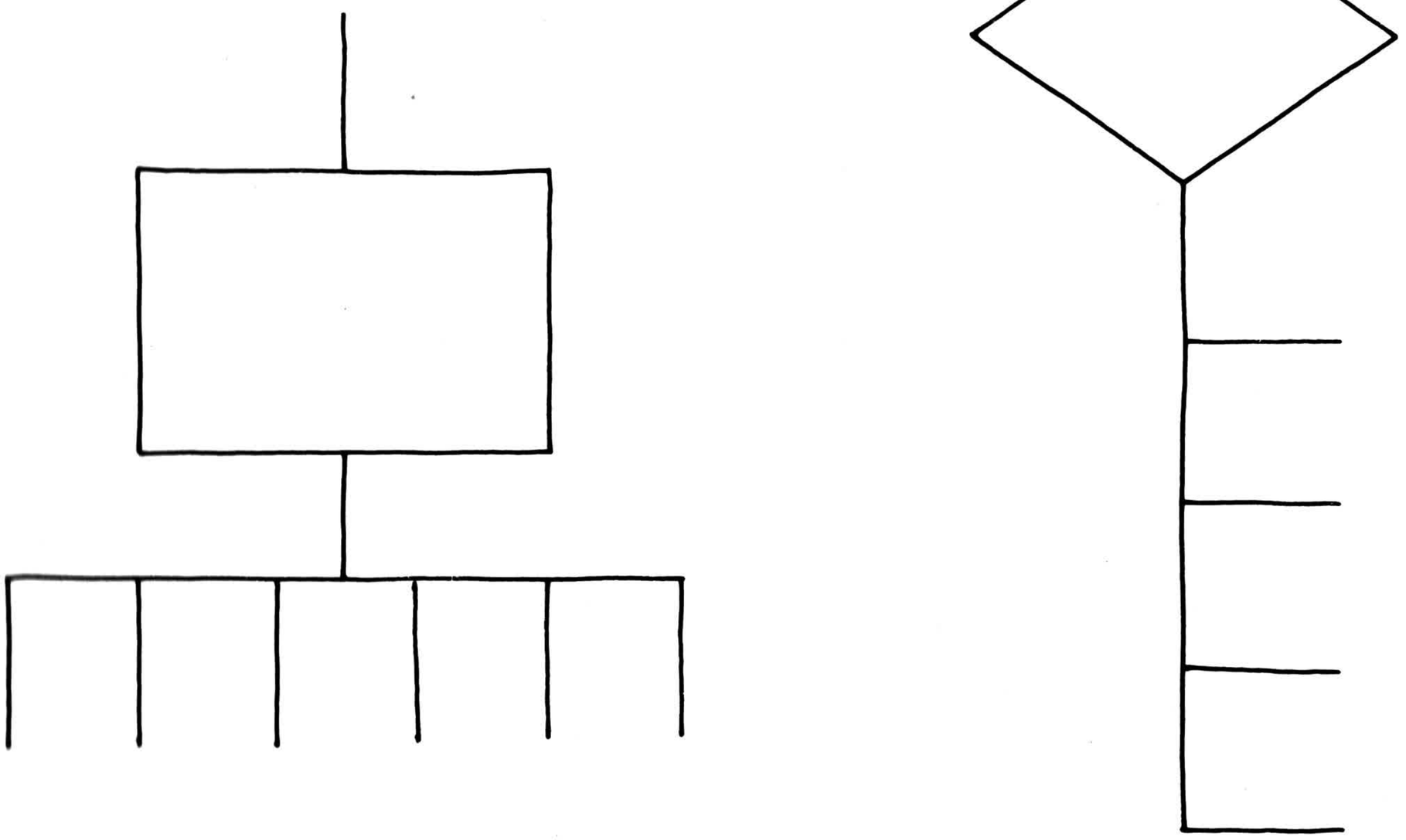
Mehrere Ausgänge

Von einem Sinnbild dürfen mehrere Verbindungen abgehen, und zwar direkt, z. B.

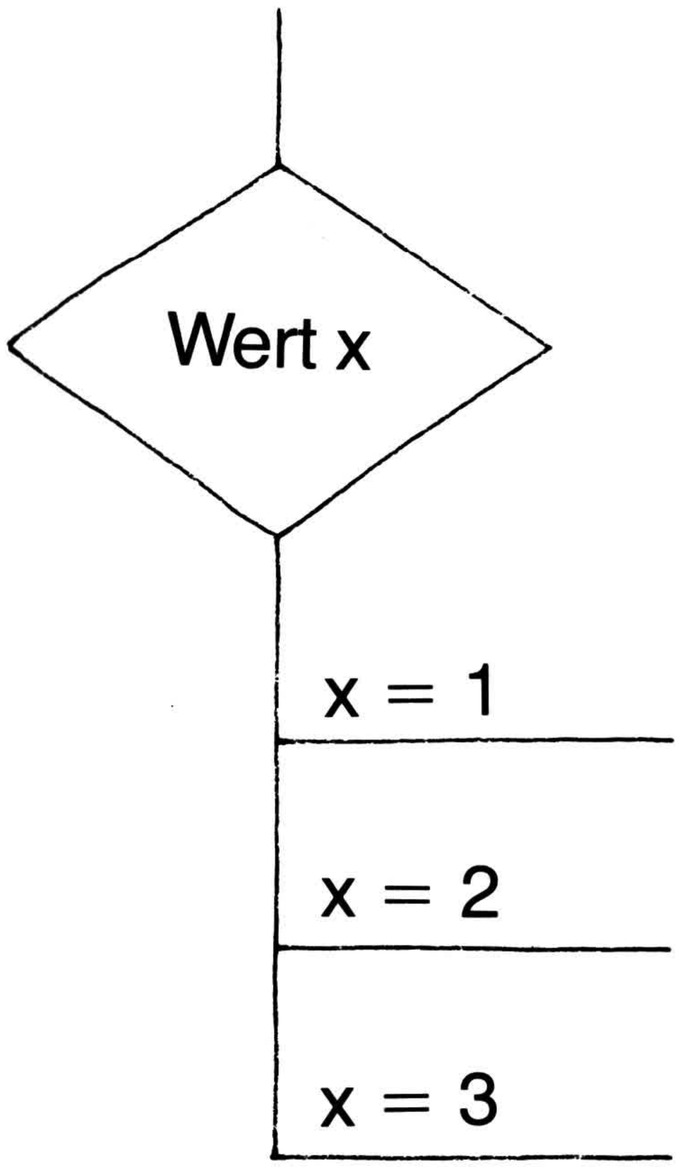




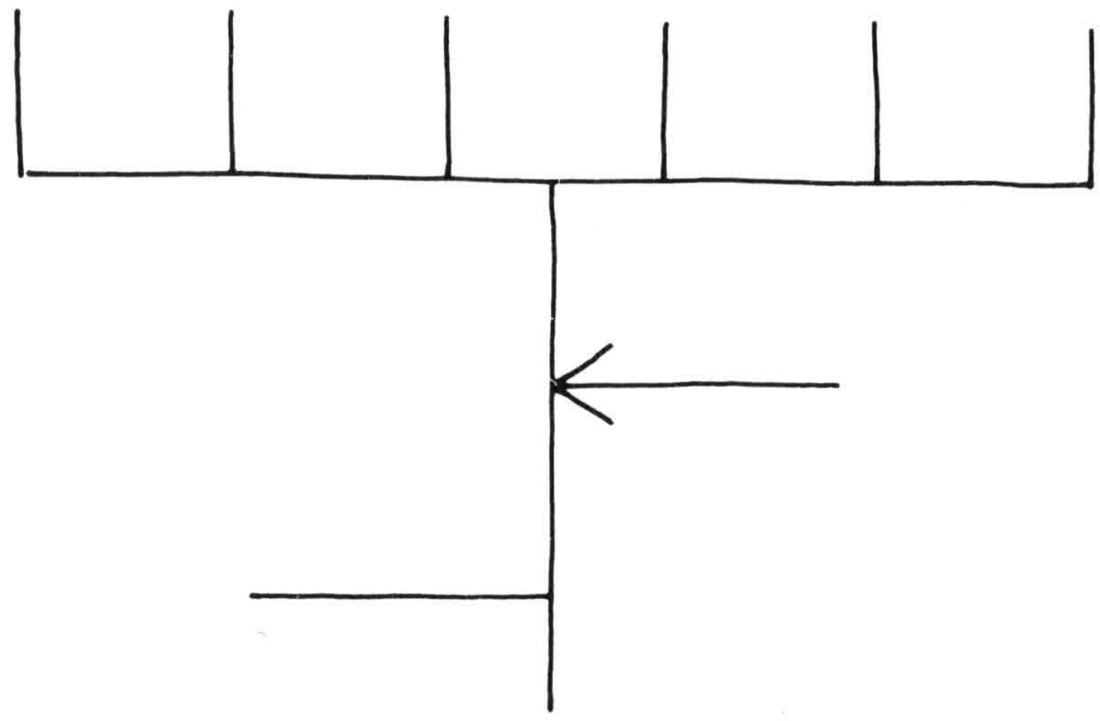
oder durch Auffächerung einer ausgehenden Linie, z. B.



Gehören zu verschiedenen Ausgängen Bedingungen, müssen sie eindeutig erkennbar sein, im allgemeinen durch Beschriftung an den Verbindungslinien, z. B.

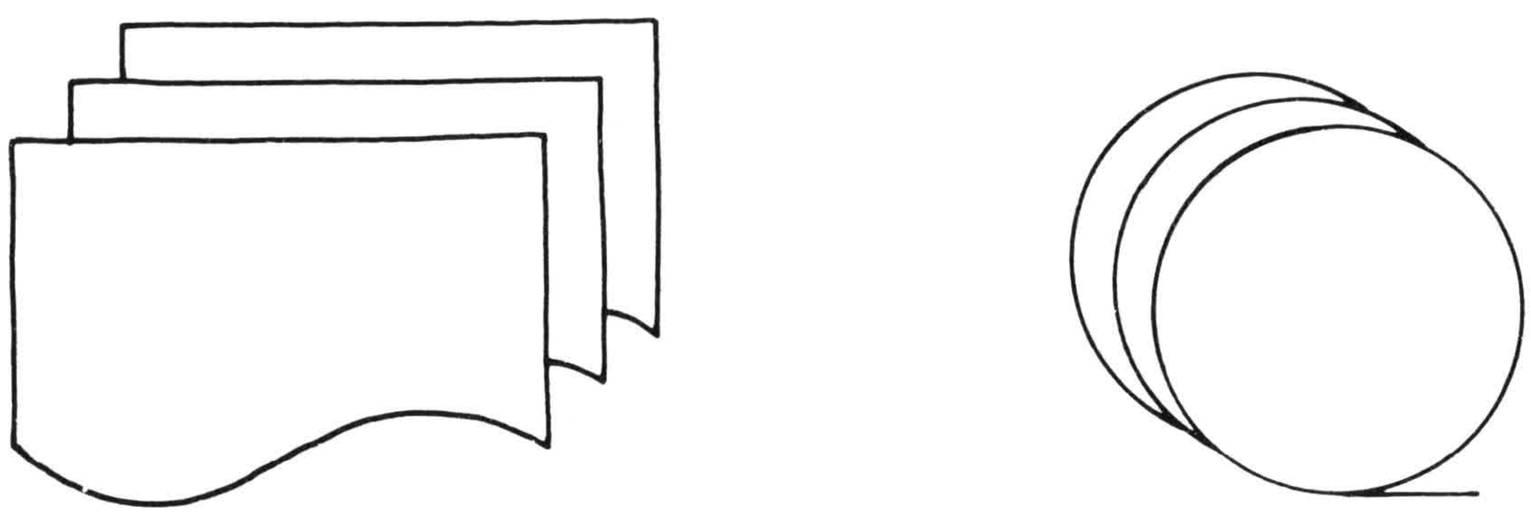


Zusammenführung[1]) **und Kreuzung von Verbindungslinien**
Führen mehrere Verbindungslinien zu einem Sinnbild, so können sie zusammengeführt werden, z. B.



Sich kreuzende Verbindungslinien sollten vermieden werden; sie stellen **keine** Zusammenführung dar.

Wiederholung gleicher Sinnbilder
Sollen im Datenflußplan oder in Programmnetzen Daten auf mehreren gleichartigen Datenträgern oder im Konfigurationsplan mehrere gleichartige Datenträgereinheiten dargestellt werden, so darf dies durch hintereinandergezeichnete gleiche Sinnbilder erfolgen, die zusammen eine Einheit bilden, z. B.



[1]) Begriff siehe DIN 44 300

Das vorderste Sinnbild ist voll auszuzeichnen, die schräg verschoben dahinterliegenden nur so weit, wie sie sichtbar sind. Soll eine Reihenfolge dargestellt werden, ist sie durch Beschriftung zu kennzeichnen.

Verknüpfung von Sinnbildern

Zur Darstellung von Daten auf Benutzerstationen bzw. von Benutzerstationen können die Sinnbilder für die unterschiedlichen Formen der manuellen, optischen oder akustischen Ein-/Ausgabe verknüpft werden. Die Verknüpfung erfolgt durch direktes Aneinanderzeichnen der Sinnbilder ohne Verbindungslinie.

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Seeßlen, Georg:
Computer kinderleicht / Georg Seeßlen; Peter Fritzsche. –
München; Wien; Zürich: F. Schneider

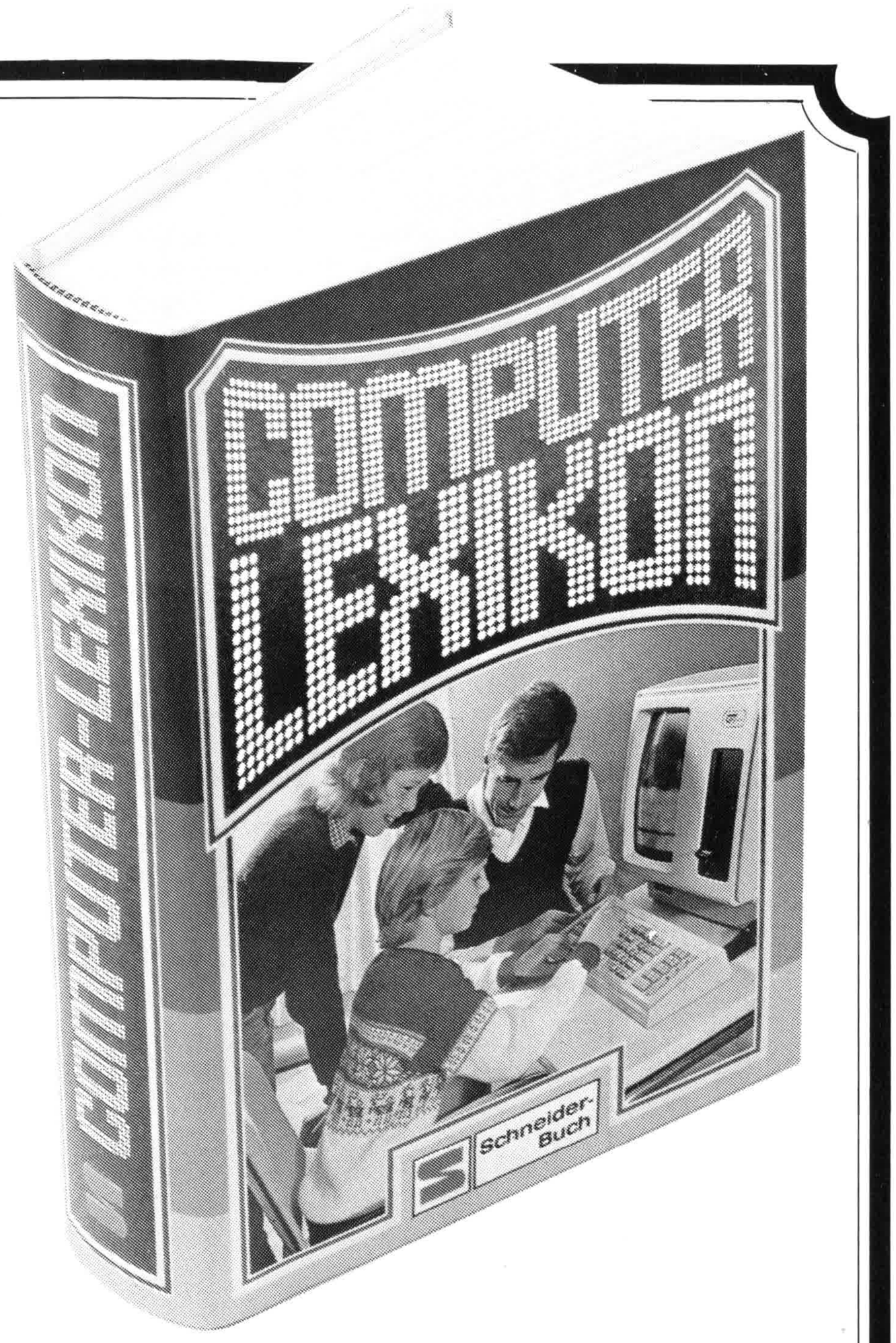
NE: Fritzsche, Peter:

Bd. 2. Lernen, spielen, programmieren. – 1985.
ISBN 3-505-09091-3
ISBN 3-505-09150-2 Buch (mit Diskette)
ISBN 3-505-09205-3 Diskette einzeln

© 1985 by Franz Schneider Verlag GmbH & Co KG
8000 München 46 · Frankfurter Ring 150 · Wien – Zürich
Lektorat/Redaktion: Thomas de Lates
ISBN Buch: 3 505 09091-3
Bestell-Nr. Buch: 9091
ISBN Buch mit Diskette: 3 505 09150-2
Bestell-Nr. Buch mit Diskette: 9150
ISBN Diskette einzeln: 3 505 09205-3
Bestell-Nr. Diskette einzeln: 9205

Weitere Bände in Vorbereitung

Erstmalig:
Mit den Adressen der Hersteller von Kleincomputern



COMPUTER-LEXIKON

Der Computer-Fachmann Klaus Jamin hat dieses Lexikon zusammengestellt. Hier findet der Anfänger und der Fortgeschrittene alles, was er sucht, klar und verständlich erklärt.

- Über 600 Stichwörter von A–Z ermöglichen den direkten Zugriff zum aktuellen Computerwissen unserer Zeit. Sie erklären das Wesentliche und geben präzise Antworten.
- Über 200 Abbildungen informieren ausführlich. Wort und Bild stellen eine Einheit dar, sie ergänzen sich optimal.
- Mit Sonderteil „Programmier-Sprache“. Damit nichts verborgen bleibt – für jeden verständlich.

Schneider-Buch

HERAUSGEGEBEN VON
F. SCHNEIDER

Wörterbuch für Hacker

Hacker sind alle, die auf ihrem PC herumhacken und sich mit ihm duellieren — also PC-Fans der ganz besonderen Art.

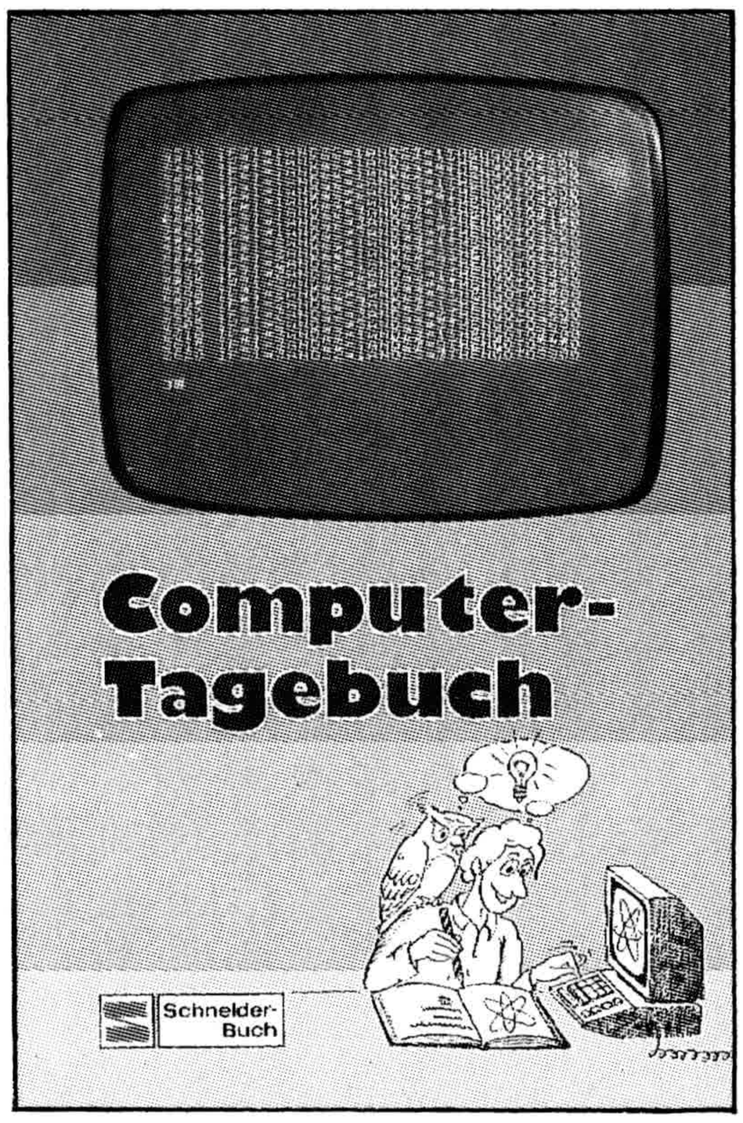
Hacker sind Neugierige in der abenteuerlichen Computerwelt. Sie lieben und sie hassen ihren PC, und sie sprechen mit ihm wie mit einem Freund, und mit ihren Freunden reden sie wie mit einem Computer. Sie haben sich eine eigene Sprache geschaffen, in der sie viele EDV-Abkürzungen verwenden.

Dieses Wörterbuch hilft, die neue Umgangssprache zu verstehen und sie richtig zu benutzen.

Warnung: Dies ist kein Computer-Lexikon!

Schneider-
Buch

HERAUSGEGEBEN VON
KLAUS JAMIN UND
NORBERT MARX



Computer-Tagebuch

Das Computer-Tagebuch schafft Übersicht und Ordnung. Ein Arbeitsmittel mit Platz für Notizen, vergleichende Einträge, Raum für Adressen, Listings und Entwürfe. Zum Entwerfen normierter Programme enthält es ein Verzeichnis der nach DIN gültigen Ablaufplansymbole und einige Programmiertips.

Als Trainingsspaß das Programm „Partnersuche" — sofort lauffähig auf dem VC 64, auf jeden anderen Typ übertragbar und mit Möglichkeiten zum Weitertüfteln.

Das Computer-Tagebuch ist ein nützlicher Begleiter durch den Computer-Alltag von Anfängern und Profis, für Fans, Hacker und solche, die es werden wollen.

Schneider-
Buch

Spannende Schneider-Computer-Bücher

G. P. JORDAN
Computer-Krimi TRM

Band 1 „Geheimprogramm Bienenstich“
Das Geheimnis des verschwundenen Wissenschaftlers

Band 2 „Projekt Gedankenschnüffler“
Was haben die Gangster mit dem Tempest-T vor?

Band 3 „Daten-Piraten“
Tom, Rick und Mona auf Crackerjagd

GEORG SEESSLEN / PETER FRITZSCHE
Computer kinderleicht

Band 1 „Wie geht das überhaupt?“
Einstieg in die Computerei

Band 2 „Lernen Spielen Programmieren“
Der Computer-Rechenkünstler

Wörterbuch für Hacker
Computerslang für Einsteiger

Hacker-Lexikon
Computerslang für Insider

KLAUS JAMIN
Computer-Lexikon
mit Hersteller-Adressen
und Sonderteil „Programmiersprache“

KLAUS JAMIN
Computer-Tagebuch
mit Programmiertips und einem witzigen Programm für den C 64

Computer kinderleicht · Bd. 2
Lernen, spielen, programmieren

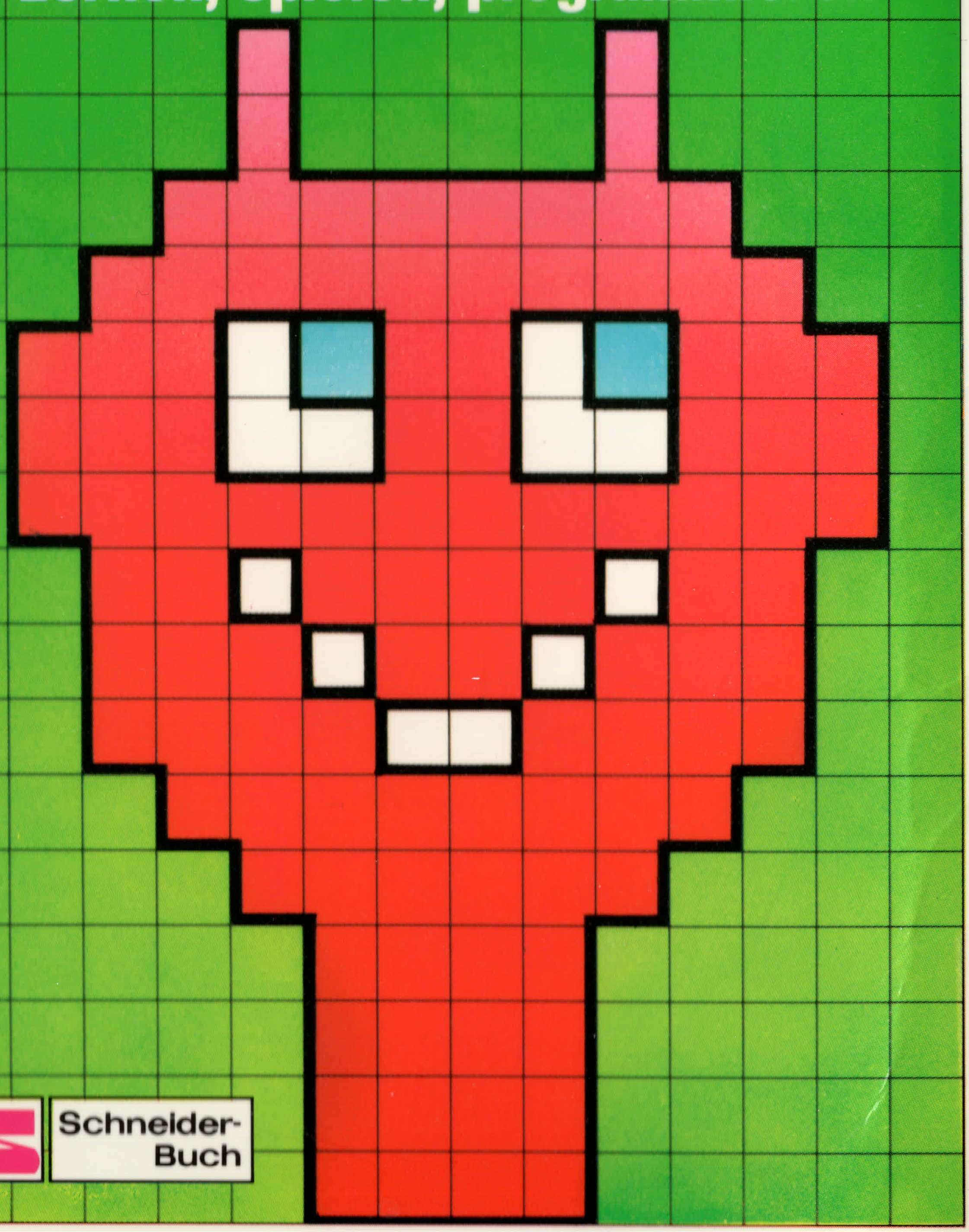
„Computer kinderleicht" – das heißt Computerei mit Spaß, BASIC lernen ohne Frust und Langeweile und vor allem: spielerisch!
Geschrieben von Computer-erfahrenen Pädagogen, getestet von Computer-neugierigen Kindern und geprüft von Computer-Profis.
Band 2 enthält das Wichtigste für den systematischen Einstieg ins Programmieren – verständlich und unterhaltsam, mit Erfolgserlebnissen von Anfang an.
Ein Buch mit zahlreichen Abbildungen und Listings, das Kinder eigenständig und weitgehend ohne Hilfe von Erwachsenen benutzen können!

9 783505 090912

ISB N 3-505-09091-3

Computer kinderleicht

Lernen, spielen, programmieren



Schneider-Buch